Using Case-Based Reasoning in Instructional Planning.Towards a Hybrid Self-improving Instructional Planner

Jon A. Elorriaga, Isabel Fernández-Castro

International Journal of Artificial Intelligence in Education (2000), 11, 416-449

Using Case-Based Reasoning in Instructional Planning. Towards a Hybrid Self-improving Instructional Planner

Jon A. Elorriaga and I. Fernández-Castro *Department of Computer Languages and Systems University of the Basque Country - Euskal Herriko Unibertsitatea 649 Postakutxa, E-20080 DONOSTIA, Spain.E-mail: jipelarj@si.ehu.es, jipfecai@si.ehu.es*

Abstract. This paper presents a new approach, based on the Case-Based Reasoning technique, which is useful to enhance Intelligent Tutoring Systems with learning abilities. Case-Based Reasoning is a recent paradigm for problem solving that has an inherent learning capability. So we integrate a Case-Based Instructional Planner within existing Intelligent Tutoring Systems (ITS) to enhance the pedagogical component with learning capabilities. In this way, the enhanced tutors combine the Case-Based Instructional Planner with a more conventional one in order to obtain a robust hybrid instructional planner. Thus, the resulting ITSs become self-improving systems that exhibit two kinds of learning: learning from memorisation and learning from their own experiences. In this paper a framework for developing case-based instructional planners together with a hybrid approach for connecting such planners to existing ITSs are presented. The methodology to perform that integration is also described. Finally, the results of the formative evaluation of a hybrid self-improving instructional planner using simulated students are pointed out.

INTRODUCTION

Instructional Planning is a central issue to develop adaptive educational software, not only Intelligent Tutoring Systems (ITSs) but also traditional educational programs and learning applications based on new pedagogical paradigms. Several definitions that justify its central role can be found:

> *"Instructional Planning is the process of mapping out a global sequence of instructional goals and actions that provides consistency, coherence and continuity throughout an instructional session"* (Wasson, 1990, pp. 43).
>
> *"Instructional Planning, in the context of machine planning for ITSs, is the process of configuring partial orderings of instructional operations that when executed are likely to result in a student achieving selected instructional objectives"* (Macmillan & Sleeman, 1987, pp. 17).

By means of Instructional Planning techniques the educational software can plan the appropriate contents and learning activities for each student. Most of the more recent instructional planners are based on production systems. However, the rule-based paradigm shows some drawbacks (Watson & Marir, 1994): the development and maintenance of rule-based system is costly, moreover it is hard to adapt rule-based instructional planners to new domains and also to situations not foreseen in advance, *e.g.* learners with special needs. One possibility to overcome these limitations is the addition of learning capabilities to the instructional planners. Although it seems to be a promising area, only a few self-improving ITSs have been built (Dillenbourg, 1989; Gutstein, 1992; Kimball, 1982; MacMillan & Sleeman, 1987; O'Shea, 1982).

The learning ability can be supported by means of machine learning and knowledge acquisition techniques. Machine learning strategies for learning from experience seem to be appropriated to be applied to the planning task. However, it is difficult to apply such inductive techniques to rule-based tutors because of the complexity of attributing the success or failure of the instructional session to concrete rules.

Here, we use a different approach to instructional planning based on the record and recovery of previous cases. So, the aim of the research presented here is to build a self-improving ITS on the basis of Case-Based Reasoning (CBR).

The basic underlying principle of CBR is to solve new problems by adapting solutions that were used to solve problems in the past (Kolodner, 1993). A case-based reasoner uses a case memory that stores descriptions of previously solved problems and their applied solutions. The CBR process consists of seeking cases that are similar to the current problem and adapting the found solutions to fit the current situation. A case-based system learns from its experiences by storing these in new cases.

The CBR technique has already been used in the field of tutoring for different purposes (Khan & Yip, 1995-b). Largely, it has been used in learning applications based on the principle of "teaching by presenting examples" (cases). In addition, some researchers (Coulman, 1991; Du & McCalla, 1991; Kolodner, 1991; Riesbeck & Schank, 1991; Khan & Yip, 1995-b) have applied this problem-solving paradigm to instructional planning in different ways, but none of them have extensively benefited from the learning capability of CBR to develop self-improving ITSs. We apply this problem solving technique to build instructional plans for ITSs. That is a Case-Based Instructional Planner coupled with a more conventional rule-based one to form a robust hybrid instructional planner with learning capabilities.

The core of this work is a Case-Based Instructional Planner (CBIP) whose learning capabilities are mainly based on the storing of the results obtained by each case, so it is able to promote the application of the cases with the best results. The approach for enhancing existing ITS with learning capabilities consists of integrating a CBIP into an existing ITS. In the resulting hybrid system, the CBIP collaborates with the original instructional planner to decide the instruction at the same time that learns from it the cases.

In order to apply this approach, we have developed a framework implemented in CLIPS (*C Language Integrated Production System*) that facilitates the construction of CBIPs for different ITSs. However, due to the necessity of using domain knowledge in some tasks (adaptation phase), its design cannot be treated deeply in a general way. In addition, we have defined a methodology to guide the development of the hybrid system with a set of guidelines to fulfil each step. Both the framework and the methodology have been designed in order to minimize the changes in the original ITS. The application of the methodology is illustrated with the construction of a hybrid self-improving instructional planner for *Maisu*, an Intelligent Tutoring System (Arruarte *et al.*, 1997).

The rest of this paper is structured as follows. The next section briefly describes the Case-Based Reasoning process. Section 3 presents a case-based approach to instructional planning. Section 4 describes the components of the Case-Based Instructional Planner in detail. Section 5 presents the way to enhance ITSs with a Case-Based Instructional Planner. Section 6 describes the application methodology of this enhancement and illustrates it with an example. Section 7 shows the results of the enhanced planner. Section 8 compares this approach to other related works. Finally, the advantages of our approach and the perspectives for future work are outlined in the last section.

OVERVIEW OF CASE-BASED REASONING

Knowledge-based Systems (KBS) form a successful area of Artificial Intelligence, whose fundamental characteristic consists of explicitly representing the domain expertise. However, this approach has met several problems (Watson & Marir, 1994):

- Knowledge elicitation is a difficult process,
- Implementing the expertise in a set of rules is a time-demanding task,
- Model-based KBSs are slow and unable to handle large volumes of information, and
- KBSs are difficult to maintain.

Case-based reasoning (CBR) (Kolodner, 1993; Watson & Marir, 1994; Aamodt & Plaza, 1994; Riesbeck & Schank, 1989) is a relatively recent problem solving and learning paradigm that seems to overcome the above mentioned problems. The basic idea of this paradigm is to

solve new problems by adapting successful solutions that were used previously. So, learning is inherent to the case-based problem solving process, when the solved problems and their solutions are directly stored in new cases.

A *case* is composed of three elements: the description of the problem, the applied solution and the obtained outcome. The set of used cases is organised in the Case Memory (CM). As the case search and matching process need to be both effective and reasonably time efficient, the representation, indexing and storing of the cases become important issues in designing case-based systems.

The first task in designing a case-based reasoner should be the identification of the relevant features of the cases. This analysis will determine the elements to represent and the indices to rapidly search through the CM. The CM should be also organised into a manageable structure that supports efficient search and retrieval methods. Two well-known case memory models are the *dynamic memory model* (Schank, 1982) and the *category-exemplar model* (Porter *et al.*, 1990).

The Case-Based Reasoning cycle involves four processes: retrieve, reuse, revise and retain which are described below (figure 1).

Retrieve: The current problem is analysed and the relevant features of the situation are used to search the CM through the indices in order to find a set of similar cases. The retrieved cases are tested with the aim of selecting the most appropriate one.

Reuse: Once a case is chosen the case-based reasoner must adapt it to fit the characteristics of the current situation. There exist two kinds of adaptation in CBR: *structured adaptation*, in which adaptation rules are applied directly to the retrieved solution, and *derivational adaptation*, which produces the new solution by re-using the method that generated the solution for the retrieved case. The adapted solution is applied to the problem in the real domain. However, the application is considered outside of the CBR process, as it is performed by an external system.

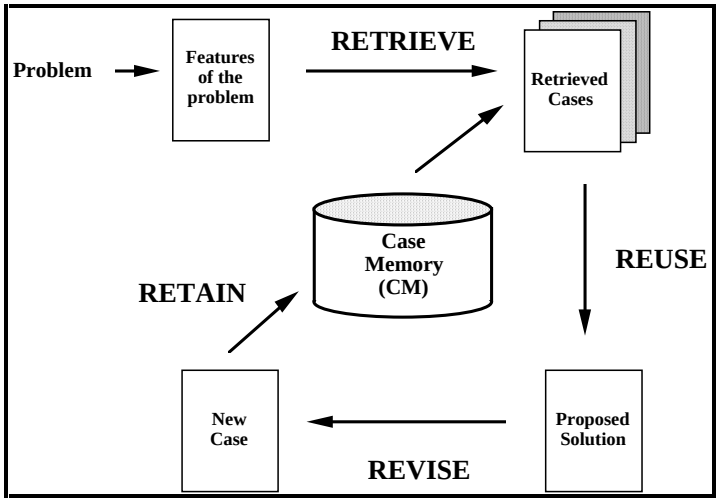

Figure 1. The CBR cycle (adapted from (Aamodt & Plaza, 1994)).

Revise: After applying the solution, the next CBR phase consists of the revision of the obtained outcome to check whether the proposed solution really solves the problem or not. If the solution was unsuccessful, the failure is explained and the solution is repaired. Therefore, there are two opportunities for learning depending on the evaluation of the results:

- When a problem is successfully solved, the data of the experience is used in the next phase to form a new case *(learning from success).*
- When the solution fails, the reason of the failure is identified, explained, and repaired. The failure and the repair are used in the Retain phase to form a case *(learning from failure).*

Retain: In this phase the current problem, the proposed solution and the outcome are stored in the CM. A new case is created extracting some data from the new experience, *i.e.* the relevant features, the applied solution and the outcome. If the solution was successful, the new case is stored to help in the future problem solving activity. If the solution failed the case is stored in the CM together with the failure and its repair, in order to avoid similar failures. The new case must be integrated into the CM and the indices updated.

CBR has been applied to a variety of systems with different purposes: knowledge acquisition, legal reasoning, diagnosis, planning *etc*. Case-based tutoring systems have been developed as well; these tutors provide instruction by presenting prototypical problems to the student such as examples or exercises.

AN APPROACH TO CASE-BASED INSTRUCTIONAL PLANNING

Instructional planning is a complex task for ITSs. Until now, ITSs have used explicit expert models of instructional planning which, in general, have been implemented as production systems. These instructional planners are difficult to build and maintain (Watson & Marir, 1994), and hardly exportable to other domains. Therefore, a planner with a learning capability is an interesting choice to overcome these difficulties.

In this research, CBR is used to develop an instructional Planner benefiting from the learning capabilities of this paradigm. The Case-Based Instructional Planner (CBIP) is the core of an approach that integrates such a planner inside existing ITSs in order to improve their performances. Thus, the CBIP is thought as a generic module applicable to several ITSs. By means of this planner the ITS learns the results of the plans, therefore, the best plans to apply in each instructional situation.

Some issues of Case-Based Instructional Planning

The CBIP is the product of an adaptation of the general CBR mechanism to fit the special characteristics of the instructional planning task. In general, the instructional plan is composed of several levels, *e.g.* (Gutiérrez, 1994; Arruarte *et al*., 1997) and thus the case-based planning process must be customised to cope with the instructional plan structure of each specific tutor. Therefore the process of generating plans is divided into a number of stages, each one builds one level of the instructional plan. Figure 2 shows a general cycle of case-based instructional planning for generating a N-layered plan. These stages are performed in a cascade, in such a way that each stage uses information obtained in previous stages. That is, each stage creates a subplan for each plan item in the previous level by retrieving old cases from the case memory and adapting them to the current situation. Once the instructional plan is completed, it is applied to develop an instructional session, whose results are *revised* by the CBIP to check whether the student achieved the expected instructional objectives and whether the didactic activities were successful. Finally, the CBIP performs an inverted process, decomposing the final instructional plan into basic plan pieces, subplans, and creates a new case for each basic plan to be *retained* in the memory of instructional plans.

This direct translation of the instructional planning process to the CBR approach meets some difficulties when the number of levels grows: building the CBIP is more complex, the process of planning takes more time and the number of cases to be stored grows. The justification to have a multi-layered instructional plan is to explicitly represent the process of building the plan itself. However, in the approach presented here, such detailed representation of the plan is not necessary. From a pragmatic point of view, the contents of the plan that are used to dispatch the teaching session are just the topics to be taught and the concrete teaching activities involving these topics. Other intermediate levels of the planner are important to extensively represent the plan and justify pedagogical decisions, but are not essential for the session dispatching. Taking into account the reasons above and in order to facilitate the generalisation of the CBIP, we use a two level (topics and instructional activities) approach for the instructional plan. Nevertheless, the CBIP can be used in the same way with more complex instructional plans if needed (Elorriaga *et al.*, 1995).

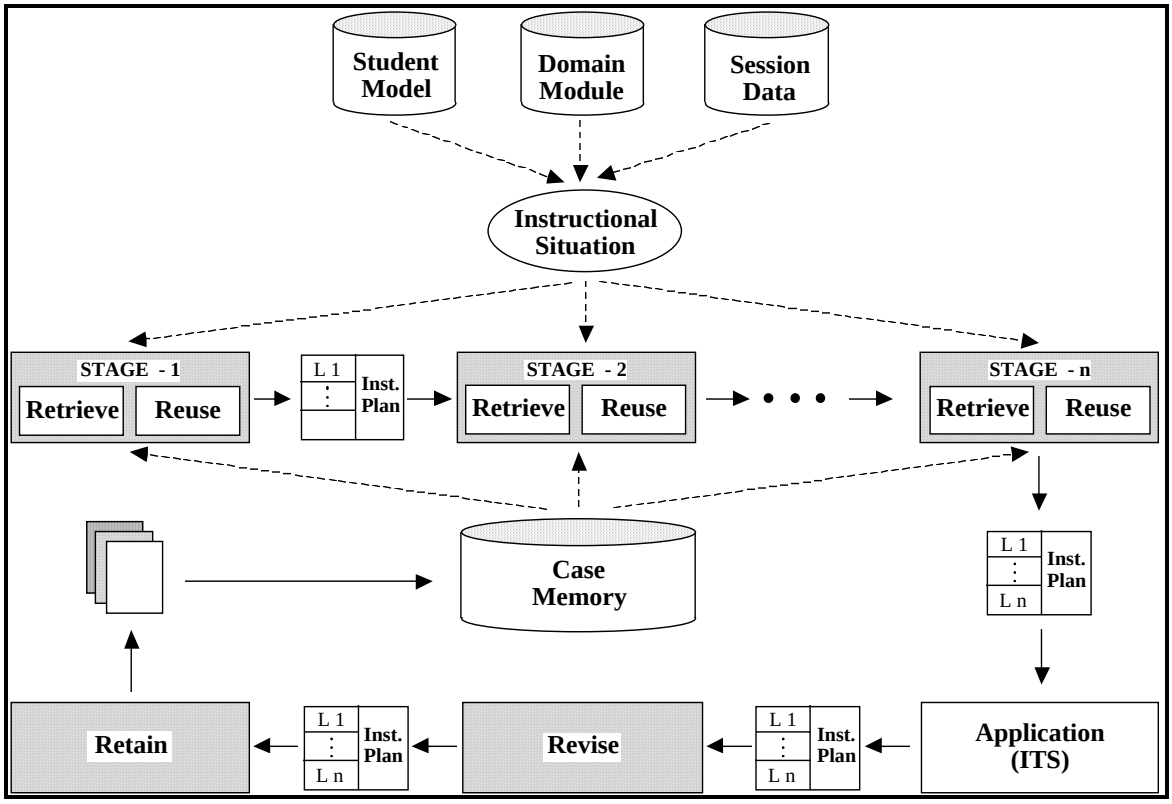

Figure 2. CBR cycle adapted to a generic N-layered Instructional Planning task.

Structure of the Case-Based instructional Planner.

CBIP is composed of three functional modules (*Generation Component*, *Learning Component*, and *Assessment Component*) and a knowledge structure (*Instructional Plan Memory*) (see figure 3). Next, we describe these components in some detail, paying special attention to the representation issues.

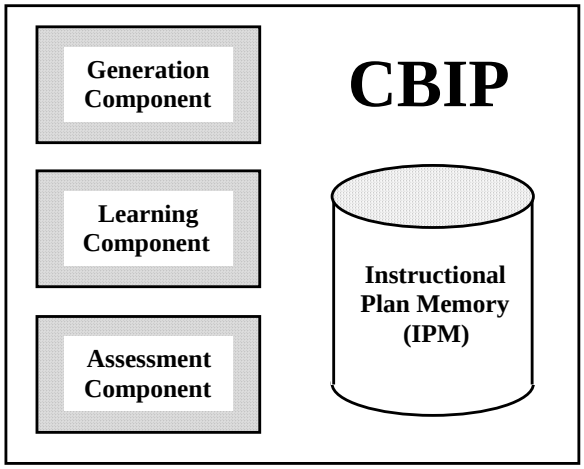

Figure 3. CBIP schema.

The **Instructional Plan Memory (IPM)** is the repository of the past teaching/learning experiences of the case-based system. It is responsible for maintaining the case set in an organised way in order to facilitate the retrieval of cases and save memory.

A *Case* defines a piece of a previously used instructional plan and contains: the context in which it was applied, the instructional plan itself or a part of it (subplan) if the plan is layered, and the results that it achieved (see table 1). The elements included in a case depend on the concrete ITS enhanced with this approach.

Table 1. Components of a *Case*.

```
Application Context
      <sequence of student related features>
      <sequence of session related features>
      <sequence of domain related features>
Instructional Plan
      <sequence of plan items>
Results of the Application
      <sequence of result values>
```

The *Application Context* describes the instructional situation in which the plan was applied and is implemented as a list of pairs `(attribute, value)`. In general, it contains information about the knowledge and preferences of the learner extracted from the Student Model, *e.g.* [`level-of-acquisition (topic), high`], domain information related to the topics included in the plan, *e.g.* [`difficulty (topic), low`], and information of the current learning session, *e.g.* [`time-from-last-session, medium`]. The Application Context is used to calculate the similarity between instructional situations and also to structure the whole memory as indices.

The *Instructional Plan* is composed of a variable sequence (or graph) of instructional plan items, *e.g.* a list of topics or a list of teaching activities; it depends on each concrete ITS.

The *Results of the Application* of the instructional plan define the last component of the case; it contains elements describing the results of the current learning session, *i.e.* the changes detected in the Student Model, *e.g.* [`change-of-level-of-acquisition (topic), low, medium`][i].

The **Generation Component** is responsible for building the instructional plans that will be executed to produce the teaching sessions; in fact, this component is the real instructional planner. The plan building process is split into several stages depending on the structure of the original instructional plan. A layer of the plan is produced in each stage, therefore the whole plan is generated in an incremental way. Each stage performs two phases of the CBR: *Retrieve* and *Reuse*.

The **Learning Component** is responsible for updating the IPM by inserting the cases that correspond to the teaching sessions already developed. So *learning* is done *directly* by incorporating in the IPM the new experiences of the CBIP for their future use, and *statistically* as the results obtained in the previous learning sessions are taken into account. This module performs its task in two phases, *Revise* and *Retain*, the last phases of CBR.

The **Assessment Component** evaluates the suitability of the elements used in the generation of the plan taking into account different aspects such as the degree of similarity between the context of application of the cases, the rating of use of the cases, or the results obtained. It is composed of a collection of adaptable heuristic formulae that estimate the suitability of the instructional plan built by the CBIP, the appropriateness of the cases retrieved from the IPM and the results of the executed plans. These estimated assessments are used: (1) by the CBIP to decide whether to use the retrieved cases, (2) by the ITS to decide whether to use the product of the CBIP, (3) by the CBIP to update the results attached to the cases, (4) and also by the search algorithms of the Generation Component.

A DEEPER VIEW INTO …

Once the basic structure has been presented, in this section we will show a deeper view of each component. As it has been explained before, the generation process as well as the learning process is split into several steps depending on the structure of the original instructional plan. In

order to simplify, this presentation is solely focussed on the treatment of a basic piece of the whole instructional plan, *i.e.* a subplan.

… IPM organisation

The organisation of the IPM has been based on the more general Schank's *Dynamic Memory Model* (Schank, 1982). In order to provide efficient retrieval methods, the memory has been organised as a hybrid network mixing characteristics from *Shared-Feature Networks* and *Discrimination Networks* (Kolodner, 1993) and exhibits properties of both of them. It produces abstractions of several cases generalising the shared features, and also discriminates between the elements included in each abstraction.

The components of the network are classified in *Primary Nodes*, *i.e.* individual experiences or abstractions of a set of experiences, and *Auxiliary Elements*, *i.e.* elements for describing and linking primary nodes. The primary nodes of this network are both the *Cases* and the *Generalised Episodes* (GE) (Aamodt & Plaza, 1994). While *Cases* represent individual experiences, *GEs* are abstractions of sets of experiences. GEs generalise the common features of a set of cases and/or GEs. The auxiliary elements are *Norms* and *Indices*.

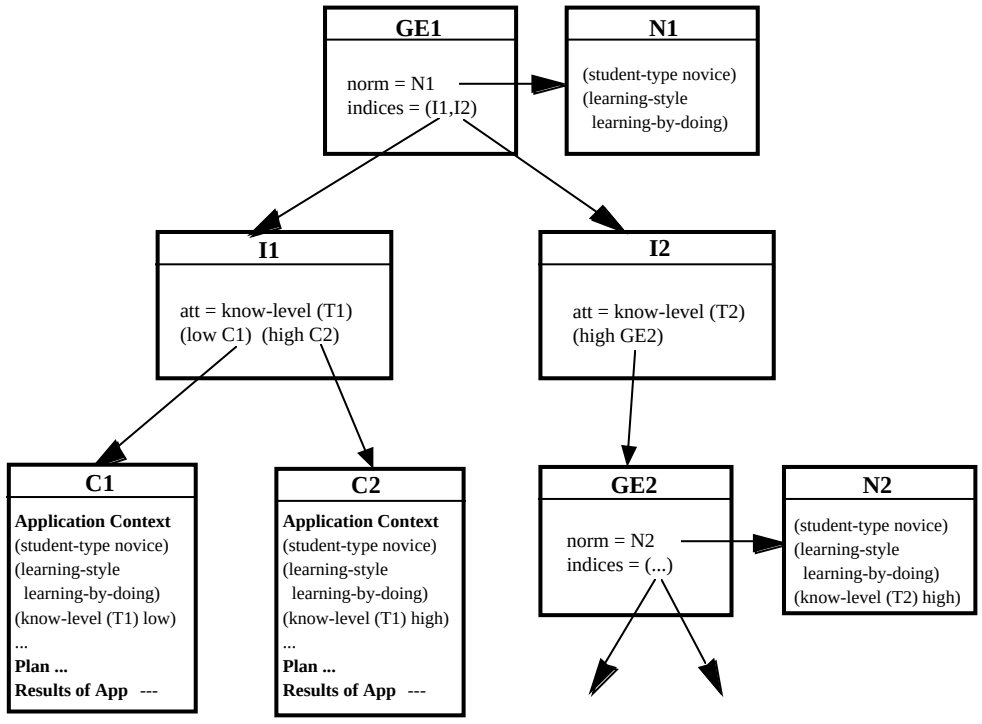

Figure 4. Fragment of the basic organisation of the IPM.

A *Norm* is attached to each GE (see figure 4) and contains the features of the application context shared by a group formed by cases and GEs; it is represented by means of a list of pairs [`attribute, value`]. The *Indices* link the network elements in such a way that each GE contains a set of indices that link it with and discriminate among its descendants. Each index is related to one concrete attribute of the application context and contains a list of pairs [`value, primary-node`]. Each pair of the index links its corresponding GE with other `primary-node` whose value for the attribute related to the index is `value`. The set of indices attached to a GE

can use different attributes to discriminate completely among the descendants. The indices can be traversed in both directions.

Figure 4 illustrates the structure of the IPM. It shows the Generalised Episode GE1 with the attached norm N1 in the root of the structure. N1 contains the attributes shared by the descendants of GE1; concretely it specifies that the instructional plans under GE1 are for novice students [`student-type novice`] whose preferred learning style is "learning by doing" [`learning-style learning-by-doing`]. GE1 is linked to the cases (C1, C2) and another Generalised Episode (GE2) by means of two indices (I1, I2). I1 uses the values of the attribute `Know-level (T1)` (student knowledge level of T1 topic) to discriminate between C1 and C2, in C1 the value is (`low`) and in C2 it is (`high`). I2 links GE1 with GE2 using the attribute `Know-level (T2)`. In the norm of GE2 (N2), it can be observed that the common attributes are the ones of GE1 plus the attribute `Know-level (T2)`.

. . . The Generation Component

The Generation Component builds the instructional plans by retrieving similar cases and adapting them to the current situation.

During the *Retrieve* phase this component obtains a set of lesson plans with a high level of similarity to the current instructional situation. The retrieved cases used to create the current plan are named *primary cases*. The similarity is measured in terms of the relevant attributes that specify the instructional situation. This task can be considered as the search of the most appropriate case through the IPM. The retrieving mechanism performs this search by looking through the memory structure and checking the matching degree of the nodes. Therefore, the retrieval of cases shows two issues: Matching and Search.

"*Matching is the process of comparing two cases to each other and determining their degree of match*" (Kolodner, 1993, pp.328). The matching method used is based on the *nearest neighbour matching* of REMIND (Cognitive Systems, 1992) that calculates the similarity among cases by using a numeric function. The *Similarity Function* (SF) is defined as the sum of the similarity measures values of each attribute in the application context of each case, multiplied by the importance of the attribute; this value is divided by the sum of the weights in order to normalise the value. The weights estimate the degree of importance of the attributes. Figure 5 shows the similarity function where w_i is the importance of the attribute *i*, *sim* is the similarity function for primitives values and f_i^1 and f_i^2 are the values for the attribute *i* in the compared cases.

$$SF(C^1, C^2) = \frac{\sum_{i=1}^{n} w_i \cdot sim(f_i^1, f_i^2)}{\sum_{i=1}^{n} w_i}$$

C^1, **C^2**: cases defined by a set of attributes f_i , $i \in [1 .. n]$
n: number of attributes of the application context
sim (f_i^1, f_i^2): similarity function for two primitive values

Figure 5. Similarity function.

The similarity function between two primitive values (*sim*) of an attribute is measured at two levels of detail, so it allows different accuracy and efficiency requirements. The coarse grain (*sim1*) is just an equality test between attribute values, and the fine grain (*sim2*) is calculated in terms of a previously defined distance metric, which depends on the type of the values. IPM stores information about the attributes of the application context to perform these calculations: the importance of the attribute ($w_i \in [0 .. 1]$) and the type of the values. Table 2 shows a simple example of matching calculation using both levels of detail and cases represented by three numeric attributes whose values range from 1 to 10.

Table 2. Example of similarity calculation.

Attributes	*Case1*	*Case2*	*Weight*	*Sim1*	*Sim2*
Attribute1	10	10	1	1	1
Attribute2	5	5	0.5	1	1
Attribute3	5	0	0.5	0	0.5
SF				0.75	0.875

The search process through the IPM is the other main issue in retrieving cases and it strongly depends on the structure of the case memory. In hierarchical memories, the search is guided by both the shared attributes of each GE and the discrimination indices. This search problem exhibits some special properties. First, the hierarchical structure of the IPM allows a heuristic search guided by the similarity function. With this purpose, the similarity function is also defined between the current application context and a GE, in this situation the function is restricted to the attributes included in the GE norm. Another characteristic of the search is that there is not an exact stop criterion, since a case sharing all of the attributes of the current instructional situation can be rarely found. Therefore, the selection criterion is based on a comparison between the result of the similarity matching formula and a heuristically established threshold.

These properties allow us to build retrieval techniques based on the classical Artificial Intelligence search methods. We have implemented a set of search algorithms that can be combined in retrieval strategies (*e.g.*: *A* search*, *Hill-climbing search*). A *Retrieval Strategy* establishes the sequence of calls to basic search algorithms and the parameters to be used in them. The parameters of a call are the algorithm identifier, the heuristic function (when it is an heuristic algorithm), the matching function, the matching threshold that a case must exceed to be retrieved and, finally, the maximum number of cases to retrieve. The knowledge engineer defines the retrieval strategy by analysing the experimental results of the search algorithms.

During the *Reuse* phase this module combines and adapts the retrieved cases to form a session plan suitable for the current instructional situation. The adaptation of the retrieved cases to the current situation is a knowledge-intensive task and therefore, needs a different treatment for each ITS in order to provide the CBIP with an effective adaptation mechanism. In order to build a generic planner, this issue cannot be defined in detail. We use a critic-based approach (Hammond, 1989) for the adaptation mechanism due to its flexibility. A *critic* is a special purpose fragment of code useful to adapt cases. They include information to identify a situation, determine the adaptation needed and the code to perform it (see table 3). The critic structure matches perfectly with that of rule-based systems. Specifically in the CBIP, the left-hand side of the critics defines the conditions in which an adaptation is needed by means of the predicates defined on the current plan, the Student Model and the domain knowledge. The right hand side contains the code of the adaptation method. (See the example in section “Methodology to build an HSIIP”). In addition, a critic includes a value that measures the adaptation degree that the code involves. The adaptation degree, a real number between 0 and 1, is set by the knowledge engineer and is used to estimate the degree of adaptation that the plan suffers during this phase.

Table 3. Schema of a critic.

```
CRITIC
IF    <conditions on the instructional situation and plan>+
THEN  < adaptation action for the instructional plan >+
Priority:              <Integer>
Adaptation-degree:     [0 .. 1]
```

. . . The Learning Component

The Learning Component is responsible for updating the IPM by inserting and updating the cases corresponding to the teaching sessions already developed. This task is performed in the last two phases of CBR *i.e. Revise* and *Retain*.

During the *Revise* phase this module evaluates each application of an instructional plan by observing the outcome of the teaching session. Each plan is evaluated as a whole and the result of the teaching session is attributed to its case; thus composing parts are not considered individually. The Learning Component analyses different pieces of information (*evaluation items*). The evaluation is carried out along two dimensions: educational and computational.

1. In the educational dimension, the relevant data concern just the Student Model. Basically, the Learning Component revises the changes in the attributes that represent the knowledge of the student and other data gathered during the session, such as, the information requests and the errors performed by the student. In addition, we think that the system's beliefs about the student should be supplemented with some feedback from the student. Therefore, the Learning Component interacts with the student after each session to gather directly the beliefs of the student about the whole instructional session as well as her/his own knowledge of the different topics worked during the session.
2. The objective of the evaluation along the computational dimension is to assess the goodness of each case as the basis to create new instructional plans. Therefore, the CBIP keeps the number of times that each case has been used to create a new plan as well as a heuristically obtained value measuring the results of the plan.

The **`result`** object represents the evaluation outcome of the applications of the case (see table 4). Hence, it stores the results of each application (**`elementary-result`**) and also maintains average information of all the applications (**`collective-result`**). Thus, the **`collective-result`** object represents the average results statistically learnt by the system.

Table 4. Objects related to the revise phase.

```
Object Result (is-a Case-component Result-object)
result-history:         <list of Elementary-result instances>
result-average:         <Collective-result instance>

Object Elementary-result (is-a Result-object)
evaluation-item-list:   <list of Evaluation-item instances>
learner:                <string>
date:                   <date>
learner-estimate:       [0..1]
global-estimate:        [0..1]

Object Collective-result (is-a Result-object)
direct-use-number:      <integer>
use-number:             <integer>
evaluation-item-list:   <list of Evaluation-item instances>
learner-estimate:       [0..1]
global-estimate:        [0..1]

Object Evaluation-item (is-a Result-object)
evaluation-attribute:   <Attribute instance>
final-value:            [0..1]
change:                 [0..1]
learner-estimate:       [0..1]
```

The revise phase produces several actions to update the IPM. First, the system decides which new cases representing the current learning session will be constructed. Only cases significantly different from the recorded cases will be created. There are two possibilities: either the new case was obtained by a substantial transformation of a recorded case (the adaptation degree attached to the applied critics exceeds a previously established threshold) or the case was

applied to a different instructional situation (the similarity degree exceeds another threshold). On the contrary, when the case is similar enough to any one of the IPM (for example when the new case was obtained by a slight transformation of a recorded case) the Learning Component attributes the effects to the primary case, so it just updates the results of that case. After revising the plan, the Learning Component builds a new case by composing: the description of the instructional situation before the session, the instructional plan and the results of the application.

Once the new cases are constructed, the Learning Component updates the **`result`** component of the cases involved in the session. For each subplan of the instructional plan an **`elementary-result`** is built, representing the result in the current session. If the subplan is represented in a new case the **`elementary-result`** is added to that case and the **`collective-result`** updated with it. If the subplan is represented by its primary case, the **`elementary-result`** is added to it and its **`collective-result`** updated. In both situations, the Learning Component updates the number of times that the primary case has been used.

During the *Retain* phase the Learning Component adds the new cases to the IPM; a reorganisation and updating of the links is produced as a side effect.

Therefore, the IPM must be provided with robust mechanisms to optimally integrate new experiences into the memory as well as to maintain its structure and properties. This involves three steps: selection of the GE in which to hold the new case, linking of the new case to the GE, and generalisation of the cases/GEs.

The selection of the GE candidate to host the new case is a search process similar to that of the retrieval. However, two differences arise: (1) the target node is a GE, (2) the algorithm can prune the network since the new case must completely satisfy the norm of the GE candidate. Variations of the basic algorithms mentioned above that take into account these characteristics have been made for this task.

The link between the GE and the new case is attained by choosing an index attribute. The selected index must fulfil some requirements: the pair `[attribute, value]` has to discriminate the new case from among the other descendants of the GE, and the `value` must be bound.

Finally, in order to maintain an optimal structure of the IPM it should be periodically checked to detect situations in which it is advisable to generalise. The IPM is tested in a process that can be executed either on-line, after each insertion, or off-line, periodically. When a set of nodes belonging to a GE and sharing a number of characteristics is detected, a generalisation is triggered; it creates a new abstraction (GE) and reorganises the links among the involved nodes.

In conclusion we want to point out the actual learning process that is carried out by the Learning Component. On the one hand, as a Case-Based system, learning is performed by storing new experiences in the IPM (Mántaras & Plaza, 1997). They can be cases produced by other planners (see next section) as well as cases that have suffered a substantial adaptation or that have been applied to different instructional situations. On the other hand, this component learns by acquiring the results of each instructional plan, this is done by representing the result of each individual application as well as the average values. The new cases and the results are used in the heuristic formulae to improve the future retrieval of cases. Thus, the planner will retrieve those cases with the best results meanwhile reject plans with bad results. Furthermore, the maintenance of average values can be considered a statistical learning strategy inside the CBIP.

. . . The Assessment Component

The Assessment Component is formed by a collection of adaptable heuristic formulae that evaluate the suitability of the following objects involved in different moments of the instructional planning process: cases retrieved from the IPM, instructional plans built by the CBIP and executed subplans. Each formula is implemented by means of low-level functions that estimate different aspects, each one is multiplied by a weight, and hence the knowledge engineer can tailor the functions by modifying its weights. Each value is normalised in the range [0 .. 1].

Let X(y) be the estimate of the aspect X and W_X the weight defined for that aspect, then we denote WX(y) as follows[ii]:

$$WX(y) = W_X \times X(y)$$

Next, we present the set of estimate formulae ordered by its complexity.

1) The function to estimate the results of a subplan, Elementary Result of a Subplan (*ERS*) (see figure 6), is used to update the **`result`** object of the corresponding cases after each session and takes into account the following aspects:

- the beliefs of the tutor about the result of the subplan in each evaluation item,
- the beliefs of the student about the result of the subplan (in the evaluation items that can be asked to the student) and also the student's overall assessment of the session.

$$ERS(S) = AEIS(S) \times WOB_L(S) \qquad ARS(S) = \frac{\sum_{i=1}^{NA} ERS_i(S)}{NA}$$

$$AEIS(S) = \frac{\sum_{j=1}^{NEI} \frac{WB_T(EI_j(S)) + WB_L(EI_j(S))}{W_T + W_L} \times W_{EI_j}}{\sum_{j=1}^{NEI} W_{EI_j}}$$

ERS	=	Elementary Result of a Subplan
ARS	=	Average Result of a Subplan
WB_T	=	Weighted Belief of the Tutor (applied to each individual feature)
WB_L	=	Weighted Belief of the Learner (applied to each individual feature)
WOB_L	=	Weighted Overall Belief of the Learner (applied to a subplan)
AEIS	=	Average of the Evaluation Items of a Subplan
NEI	=	Number of Evaluation Items
NA	=	Number of Applications of the case
EI_j	=	Evaluation Item j
S	=	Subplan
W_X	=	Weight related to factor X

Figure 6. Elementary Result of a Subplan (*ERS*) heuristic formulae.

The results of each application of a subplan are stored individually and the average values are also maintained in the case. Figure 6 shows the formula used to estimate the Elementary Result of a Subplan (*ERS*) and the Average Result of a Subplan (*ARS*).

$$RCE(C) = ARS(S(C)) \times RCAF(C)$$

$$RCAF(C) = \frac{WSF(C) + WRF(C) + WOB_L(C)}{W_{SF} + W_{RF} + W_{OB_L}}$$

RCE	=	Retrieved Case Estimate
ARS	=	Average Result of a Subplan (see figure 6)
RCAF	=	Retrieved Case Appropriateness Factor
WOB_L	=	Weighted Overall Belief of the Learner (applied to a case)
WSF	=	Weighted Similarity Factor
WRF	=	Weighted Reuse Factor
C	=	Case
S (C)	=	Subplan attached to the case C
W_X	=	Weight related to factor X

Figure 7. Retrieved Case Estimate (*RCE*) heuristic formulae.

2) The function to estimate how appropriate the retrieved cases are, Retrieved Case Estimate, (*RCE*) (see figure 7), is used to select primary cases from the IPM and must take into account:

- the degree of similarity between the current instructional situation and the application context of the retrieved cases,
- the average result of previous applications of the retrieved case, it includes the beliefs of the tutor and the learner.
- the use ratio of the case.

Figure 7 shows the set of formulae used to define a *RCE*. The main formula is composed of two factors: (1) the Retrieve Case Appropriateness Factor (*RCAF*) that calculate the overall assessment of a case, and (2) the Average Result of the Subplan (*ARS*) that takes into account the previous results.

$$CPE(IP) = \frac{\sum_{i=1}^{NL} CLE(L_i(IP)) \times W_{L_i}}{\sum_{i=1}^{NL} W_{L_i}} \qquad CLE(L) = \frac{\sum_{i=1}^{PCN_L} RCE(C_i(L)) * CPF(C_i(L))}{\sum_{i=1}^{PCN} CPF(C_i(L))}$$

$$CPF(C) = \frac{WTF(C) + WIF(C)}{W_{TF} + W_{IF}}$$

CPE	=	Created Plan Estimate
NL	=	Number of Levels in the Instructional Plan
CLE	=	Created Level Estimate
PCN	=	Number of Primary Cases
RCE	=	Retrieved Case Estimate (see figure 7)
CPF	=	Created Plan Factor
WTF	=	Weighted Stability Factor
WIF	=	Weighted Importance Factor
IP	=	Instructional Plan
L, L_i	=	Levels of the Instructional Plan
C, **C_i**	=	Cases
W_X	=	Weight related to factor X

Figure 8. Created Plan Estimate (CPE) heuristic formulae.

3) The function to estimate how appropriate the instructional plan proposed by the CBIP is, Created Plan Estimate (*CPE*) (see figure 8), must take into account several issues for each primary case used to construct the plan:

- the estimate value of the retrieved case
- the degree of stability, defined as the inverse of the degree of adaptation suffered by the plan of the retrieved case, and
- the degree of importance of each primary case in the final plan.

Figure 8 shows the set of formulae used to define *CPE*. The main formula, Created Plan Estimate (*CPE*), is a generalisation of *RCE* (see figure 7); it takes into account that a plan is composed of a set of primary cases.

THE HYBRID SELF-IMPROVING INSTRUCTIONAL PLANNER (HSIIP)

Besides the CBIP, we present a way for enhancing ITSs with learning abilities that consists of the integration of the CBIP into an existing ITS. This hybrid approach can be applied to different ITSs, provided that they are guided by an explicit instructional plan; *e.g.* (Gutiérrez,

1994; Fernández-Castro *et al*., 1993; Wasson, 1990; Murray, 1990; MacMillan & Sleeman, 1987). Let's assume an ITS with the classical architecture: Tutor Module, Domain Module, Student Model and Interface. Concretely, the CBIP must be included into the Tutor Module whose functionality can be seen as divided in two components a *Didactic Instructor* and a *Didactic Dispatcher* (the former plans the teaching session and the later executes it). The enhancement of the ITS extends its basic structure with the components of the CBIP which are introduced taking into account their functionality as shown in figure 9.

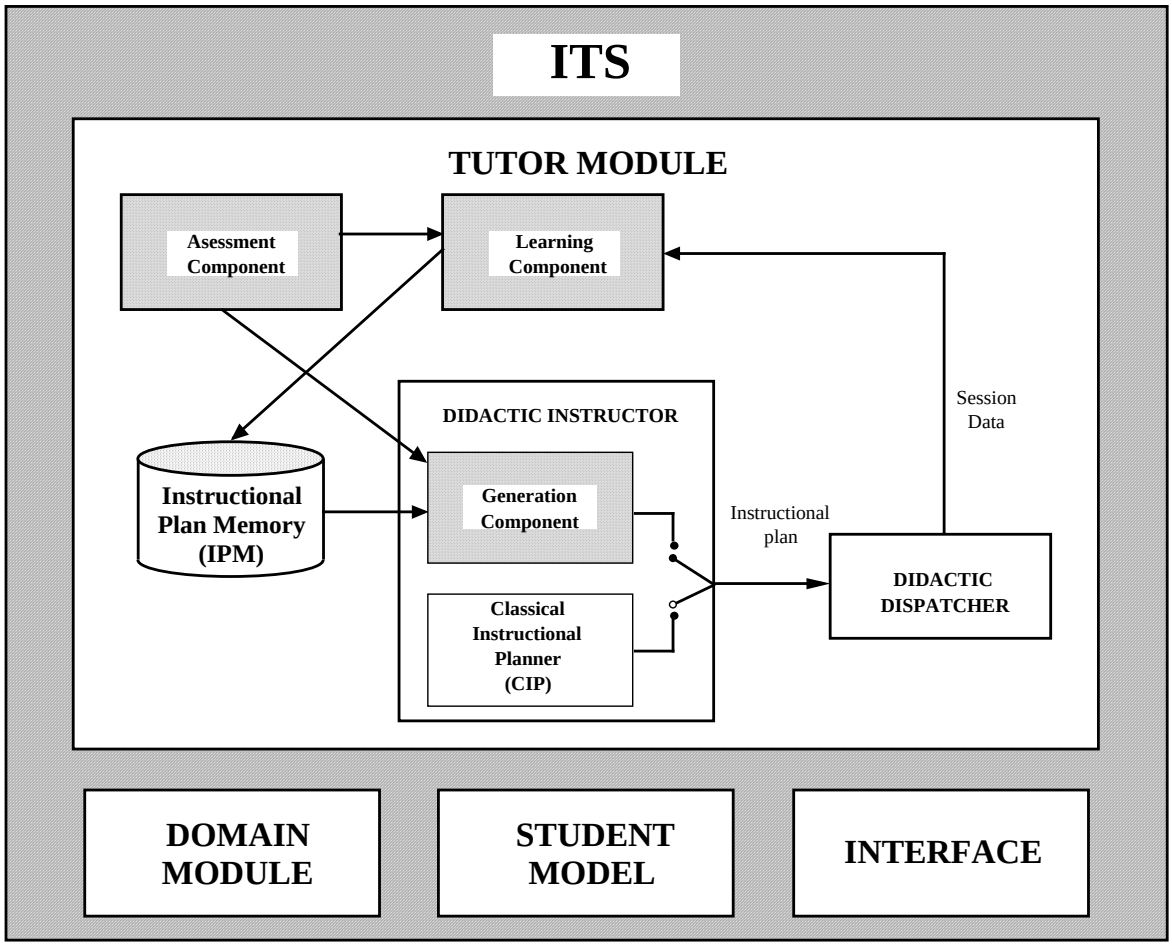

Figure 9. Architecture of an ITS enhanced with the CBIP.

Therefore, the objective of the whole project is twofold, to build an object-oriented framework for case-based instructional planning that facilitates the construction of CBIPs, and to establish a methodology for integrating the new planners within ITSs.

An ITS extended with the CBIP works as follows. The Didactic Instructor plans the session supported by both the Classical Instructional Planner (CIP) and the Generation Component; in addition it is responsible for deciding which plan of those proposed by both planners will be used. The Didactic Dispatcher carries out the session: it refines the plan in basic actions and executes them. The Learning Component monitors the development of the session and evaluates and stores the new cases in the case memory; therefore, this module treats plans built both by the Generation Component of CBIP and by the CIP.

Initially the case memory of the CBIP is empty and the planning process is performed by the CIP of the ITS (*e.g.* a dynamic opportunistic planner). In this first stage (CBIP in training phase), the Learning Component monitors the input and output of the CIP and creates new cases representing each new instructional session that are included in the IPM. With this behaviour, the Learning Component performs a kind of learning by memorisation that gathers the plans built by the CIP together with other information and store this knowledge in the format of the CBIP. When the case memory contains enough cases for the CBIP to work, it will reuse them to

generate instructional plans. However, if the CBIP fails in building an appropriate plan (the estimate of the plan built by the CBIP does not exceed a pre-established acceptance threshold), the ITS can use the plan proposed by the CIP (CBIP in co-operation phase). Combining both planners, we get a sound system that is able to learn from its experience and consequently overcomes the gaps in the case memory using the original planner.

During the development of a session, the Tutor monitors the performance of the student in order to both update the Student Model and to check if the plan is still appropriate. The tutor uses certain didactic actions (*e.g.* exercises, tests, and so forth) to get feedback from the student in order to infer changes in the Student Model. The information gathered during the session on the student's performance and the inferred level of acquisition of the topics are used to evaluate the suitability of the plan. At this point, the need to dynamically modify the instructional plan (re-planning) is clear. If the tutor decides that the current plan is not valid for the rest of the session (*e.g.* when the student makes a serious error while performing an exercise) then the instructional planner should consider the current situation and modify the plan in order to take into account the new conditions. The extended system is based on the original capabilities of the system, therefore if the original tutor is not able to replan, the enhanced one will not be able to either. The objective of this research is not focused on the dynamism of the planner but in its learning ability. However, the approach proposed to replanning is to treat the unforeseen situations by means of local instructional plans, also generated following the same hybrid instructional planning approach.

A Framework for case-based instructional planning

In order to design a generic module of case-based instructional planning it is necessary to identify the general elements that participate in the task. Therefore, the first task to construct this framework was to analyse the components and processes of case-based instructional planning that could be generalised. After analysing the process of case-based instructional planning, we generalised the representation schema of the IPM, and the *retrieve*, *retain* and *revise* phases on the basis of the CBIP model presented above. Therefore, the framework includes a set of general methods for retrieving cases, incorporating new cases and evaluating the results of the instructional plans. However, the adaptation of the retrieved cases to the current situation is a knowledge-intensive task and therefore, would need a different treatment for each ITS in order to provide the CBIP with a powerful adaptation mechanism.

In order to generalise the tasks of CBIP, it must explicitly and separately represent some knowledge about the concrete ITS that is used, for different purposes, in the CBR cycle. For example, the system must know what the appropriate attributes are for indexing the IPM, what are the attributes that should be analysed during the evaluation phase, and so on. Therefore, a structure named **CBIP Meta-Knowledge (CMK)** represents the ITS knowledge that is needed to be included in the CBIP framework to get an instantiation of the CBIP for a concrete ITS.

The CBIP Framework is a generic module for case-based instructional planning that must be adapted to fit the special characteristics of each individual ITS. It has been designed and implemented as a reusable component that contains a collection of objects and the common methods of the case-based instructional planning process. The CBIP framework has been implemented using CLIPS (*C Language Integrated Production System*).

The CBIP framework (see figure 10) is composed of a set of objects and methods for Case-Based Instructional Planning and its organisation includes the CBIP plus the CMK. Next, the components of the framework and their relationships with the CMK are pointed out.

Concerning the IPM, the framework incorporates its organisation schema in a set of general object definitions.

The Generation Component contains a set of generic methods to retrieve past experiences; it uses information of the CMK to determine which attribute to take into account in the search process. The framework includes several types of values (numerical values, discrete values and exclusive values) and provides different distance functions to define the matching functions. As it is said above, the reuse phase, *i.e.* adaptation of the retrieved cases, demands detailed knowledge and it is difficult to generalise. Nevertheless, in order to facilitate this task, the framework adopts the critic-based adaptation approach (Hammond, 1989) and incorporates several basic actions to change the plan, *e.g.* deletions, substitutions, insertions and

interchanges. In addition, a *local search*[iii] based on an abstraction hierarchy of the items of the instructional plan has been implemented to select candidate items for performing substitutions into the plan.

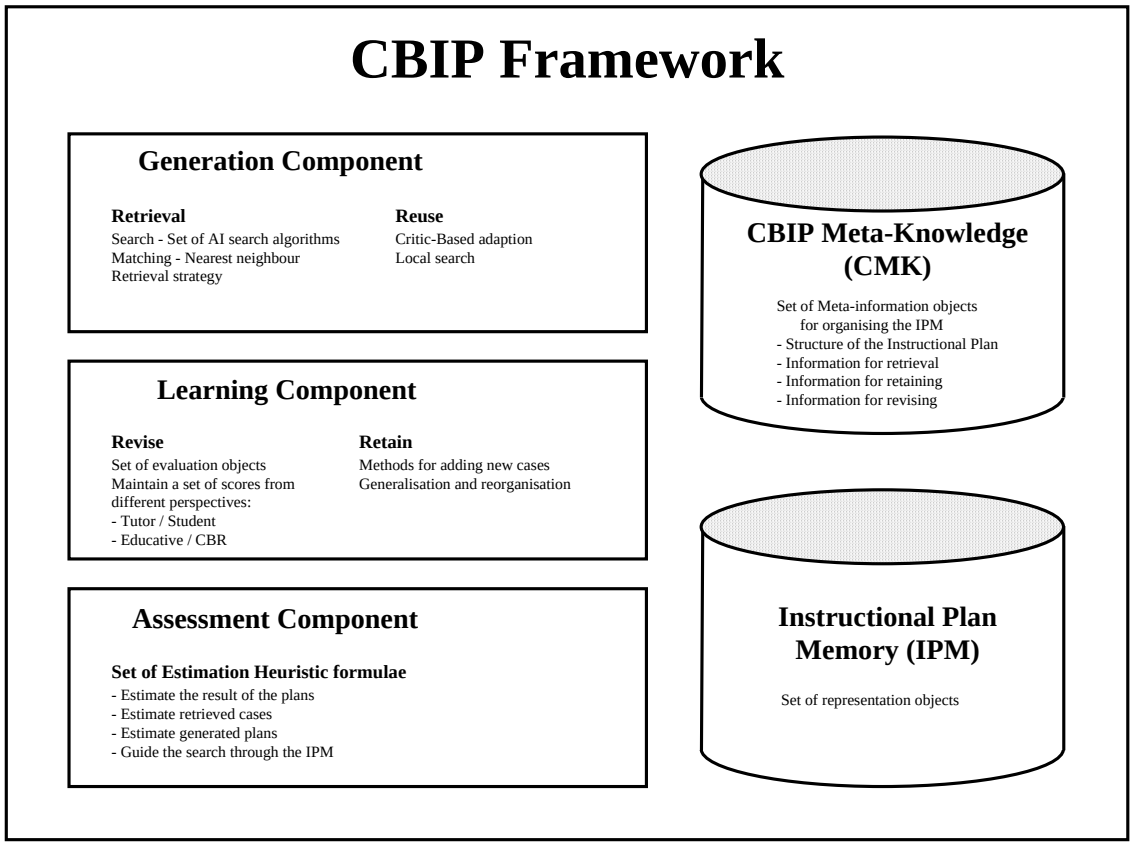

Figure 10. Framework of CBIP.

The Learning Component includes the methods to dynamically add new cases to the IPM and reorganise it by selecting the appropriate indices after consulting the CMK. Moreover, the evaluation process has been implemented as a general method based on the CMK that states which features can be evaluated and even whether it is possible or not to ask the learner about them.

The Assessment Component contains the set of estimation heuristic formulae used in the other modules that can be adapted and enhanced, changing the weights of the factors and adding new ones.

A deeper view into CBIP Meta-Knowledge

Figure 11 shows an object-oriented view of the CMK. It is composed of a set of related objets whose central object contains information about the level structure of the instructional plan (**`IP-structure`**).

Each level (**`Level-structure`**) holds the information to perform planning at that level. On the one hand, it contains information about the items of the instructional plan (**`Plan-item`**) at that level. This information includes the type of the plan items of the level, the plan item list and also an abstraction hierarchy (**`Hierarchy-node`**) of the items (see hierarchy examples in figures 13 and 14) when it is available. This hierarchy will allow using local search in the adaptation phase. On the other hand, the CMK maintains information of the attributes (**`Attribute`**) that are relevant to the planning in each level. The **`Level-structure`** object contains:

- a list of the attributes that are useful in the retrieval phase (`retrieve-att-list`),

- an ordered list of the features that are used as indices when retaining new cases (`retain-att-list`),
- those attributes which can be evaluated *i.e.* evaluation items (`revise-att-list`) and also
- the attributes that can be asked about to the student (`learner-att-list`).

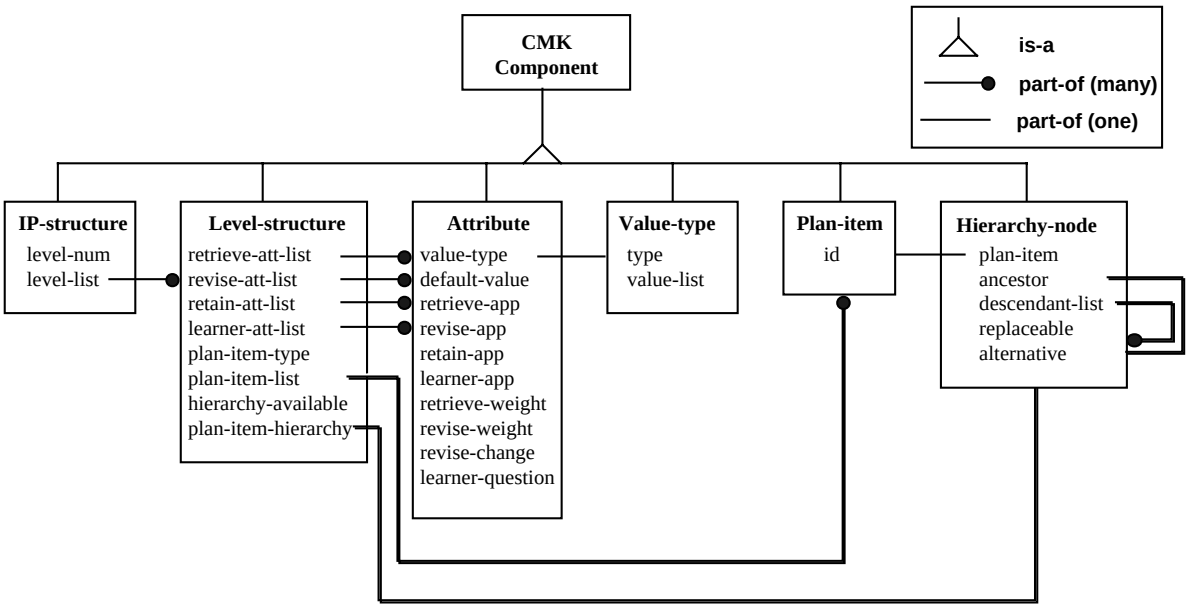

Figure 11. Object structure[iv] of the CMK.

The **`Attribute`** object represents the information known about each attribute, which is relevant to the instructional planning process. It contains:

- the allowed values (**`Value-type`**),
- the default value,
- whether the feature is appropriate to be used for retrieving cases (`retrieve-app`) and if this is appropriate, its weight in the matching formula (`retrieve-weight`),
- whether the feature can be evaluated (`revise-app`) and if this is appropriate, its weight (`revise-weight`) in the formulae of the Assessment Component,
- whether the attribute can be use as an index to link new cases /EG; and finally,
- whether any question can be asked to the learner about the feature (`learner-app`) and the question itself (`learner-question`).

The information included in the **`Value-type`** object is extremely important in order to calculate the similarity between the values of the attributes.

The **`Hierarchy-node`** is the basic piece of the hierarchy of plan items and contains:

- The plan item,
- The ancestor of the node in the hierarchy,
- The descendants of the node in the hierarchy,
- Whether the item can be replaced, and
- Whether the item can be used to replace other items.

METHODOLOGY TO BUILD AN HSIIP

In this section the procedure to integrate the CBIP into an existing ITS will be presented and illustrated with an application example.

The method to integrate CBIP into an existing ITS requires a hard knowledge engineering task in which the ITS is deeply analysed and the CBIP framework is customised to fit it. This task can be performed in different levels of detail, but obviously, the deeper the analysis, the better the CBIP produced. We also point out that the Hybrid Self-Improving Instructional Planner (HSIIP) is based on the characteristics that the original ITS exhibits.

The method for carrying out the integration of the CBIP into the ITS is performed in four steps:

- **Analysis of the ITS**. First the original ITS must be analysed in order to extract information needed to customise the CBIP. This is the most important task and requires the biggest effort by the knowledge engineer. This information concerns: the structure and components of the instructional plan, the features taken into account in the Student Model (*e.g.*: knowledge of the learner, preferences, *etc*.) and also other information relevant to represent the instructional context, such as duration or type of the lessons.
- **Adapting CBIP**. In the second step, the CBIP framework must be adapted to the characteristics obtained in the previous phase. The main objective of this phase consists of representing the information about instructional planning gathered, in the first phase, in the CMK structure.
- **Integrating CBIP**. The third phase consists of connecting both planners in the Didactic Instructor.
- **Testing**. Finally, the testing of the enhanced system must be accomplished.

This procedure is not a linear process since some problems may be detected while testing, and this circumstance may imply going back into the procedure and changing the target HSIIP.

Next, the steps of the methodology are described focussing especially on the analysis and adaptation tasks.

Step 1: analysis of the ITS

This first task consists of the analysis of the target ITS in order to extract those characteristics relevant to the instructional planning process that will be used to tailor the framework for the tutoring system. The focussed issues are: the structure and components of the instructional plan, the features of the Student Model used by the original planner to generate the instructional plan, and the features that represent the progression of the learning. So the knowledge engineer must analyse the instructional plan, the Student Model, and the original instructional planner. The main result of this phase is the knowledge to be represented in the CMK that includes the specification of three issues:

- The **level structure** of the instructional plan. By default, the CBIP framework works with a two level plan, assuming that these two layers can be found in more complex plans as well. However, more complex plans can also be treated in all their levels assuming a larger adaptation of the framework.
- The **items of the instructional plan** at each level. Furthermore, if the knowledge engineer is able to organise the items in a hierarchical way, this structure can be used to perform local search in the case reuse phase. The nodes of this hierarchy represent the items and are labelled with two characteristics: whether the item can be substituted and whether the item can be used to replace others. This hierarchy might not be present in the original tutor, but it could be added after classifying the items.
- The **attributes** relevant to the instructional planning process, *i.e.* those attributes taken into account by the ITS when planning the sessions as well as the collection of attributes that represent the measures to be evaluated after an instructional session. Most of these attributes are features of the Student Model; concretely, the attributes that represent the

knowledge supposed to the student are essential both to create the plan and to evaluate its effects. Once the features relevant to the general process of instructional planning are gathered, they must be studied in relation to each level of the instructional plan since, for each level, the set of attributes that must be taken into account are different. The study of the features relevant to each layer of the instructional plan must include the following aspects:

1) *Are the features useful for retrieving cases?* This information is obtained by analysing the original planner mechanism of the ITS and the Student Model. For instance, if the original planner is implemented as a production system, the knowledge engineer must inspect the condition elements of the rules. After this examination, the knowledge engineer obtains:
   - The set of features that are important to decide which cases are similar; each feature receives a weight according to its degree of importance. As we stated before, the algorithms to calculate the degree of matching between two instructional situations use the weights attached to each feature. In addition, for each attribute, the type of the allowed values must be extracted because they are needed for the matching functions.
   - The list of the features that will be used to select the indices that organise the IPM. The features are ordered taking into account the degree of appropriateness to be used as indices. This is directly related to the issue above, but not all the features useful for retrieval can be used as indices. For example, numeric attributes are not appropriate, because they have a wide set of allowed values, and it is difficult that a case exactly matches the values of the index. Therefore in order to use these type of values as indices the search algorithm would be more complex and, consequently, less efficient. Nevertheless these attributes can be used to calculate the similarity of instructional situations.
2) *Can the features be used to evaluate the results of the session?* This information must be obtained by extracting the features from the Student Model that represent the consequences of the learning sessions, *e.g.* the level of acquisition of the concepts, the misconceptions, *etc*. It provides the CBIP with the basic knowledge to statistically learn the results of the plans. The knowledge engineer must obtain:
   - The set of features in which the ITS stores the results of the learning session; each feature must be weighted according to its degree of importance. The Assessment Component uses these weights to estimate the global evaluation of the session. For some features only the final value is relevant, while for others the change of the value within the session is also important.
   - The set of features that the HSIIP can ask to the student. S/he enriches the system with his/her own opinion about some evaluable features.

The structural knowledge gathered in this step will serve to adapt the framework by implementing the ITS-dependent knowledge within the CMK.

The example: Analysing the target ITS

The presentation of the methodology of application is interspersed with an illustrative example. Concretely, we will apply this hybrid approach to *Maisu*, a tutor on the derivatives domain (Arruarte *et al.*, 1997). *Maisu* bases the development of learning sessions on an explicit instructional plan and represents the knowledge and learning preferences of the student in a dynamic Student Model. Therefore it meets the requisites to apply this approach.

The instructional plan contains the concepts and activities to be performed during the lesson; it is generated adaptively to each student and dynamically modified according to his/her requirements. On the one hand, the tutor adapts the instructional plan in order to treat the errors detected in the student's performances. On the other hand, as a mixed initiative tutor, the instructional plan is modified each time the student makes a request.

The analysis of the ITS will bring out the relevant aspects to tailor the CBIP system. Next, the results of the analysis concerning the three main issues are presented: structure of the instructional plan, items of the instructional plan and relevant attributes in the learning process.

Structure of the instructional plan

The Didactic Component of *Maisu* generates a five-layered instructional plan (Arruarte *et al.*, 1997) (see figure 12). The first level is constituted by a sequence of Basic Learning Units (BLU) which represent the fragment of domain knowledge to be learnt. The second level defines the Instructional Objectives (IO) that are the cognitive skills that the tutor wants the student to achieve throughout the training session. At the third layer, the plan contains the Cognitive Processes, which correspond to the mental activities that happen in the student's thought process. The fourth layer includes the Instructional Events occurring in a learning situation. Finally the last level, Instructional Actions (IAs), stands for the concrete instructional actions that the ITS carries out to teach. The IAs can be classified in two categories: the Didactic Actions (DAs) are the teaching activities that the tutor programs for the session (*e.g.* explanations, exercises *etc*.), and the Operative Actions (OAs) define the link activities between Didactic Actions for guiding and motivating the trainee during the accomplishment of an IO.

This five-layered instructional plan is a complete and explicit representation of the process of selecting the instructional actions to develop in a learning session. However, solely from the viewpoint of the instruction dispatching, the main levels are those of BLUs and IAs. These two levels indicate which concepts will be learnt in the session and the Instructional Actions planned to attain them. As the CBIP does not need to represent the complete information used to generate the instructional plan, but only the information needed to guide the learning session, the analysis will only focus on these two levels.

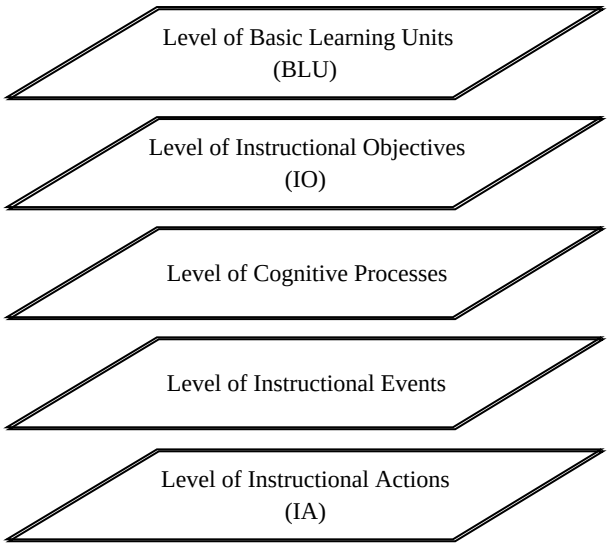

Figure 12. Structure of the Instructional Plan of the target ITS, *Maisu* (Arruarte *et al*., 1997).

Items of the instructional plan

Once the plan levels have been identified, the knowledge engineer has to collect the items from each level and, if possible, arrange them in a hierarchical structure that can be used to perform local search.

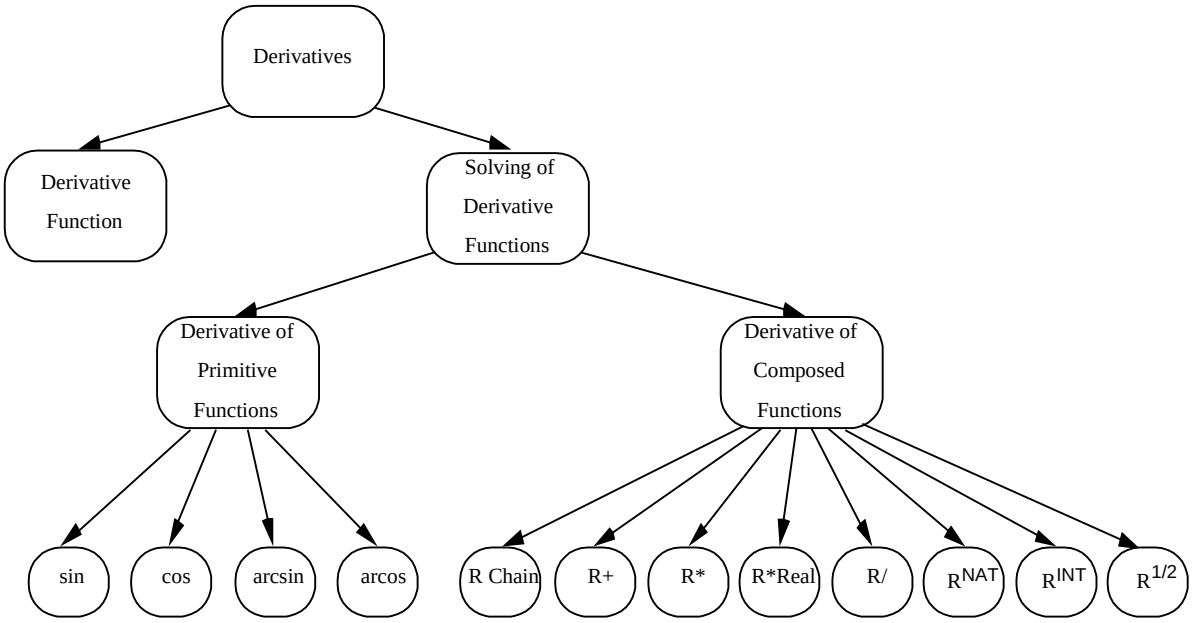

Figure 13. Schema of the domain of derivatives in *Maisu*.

Concerning the BLU level, the items are already organised in *Maisu* with some relationships. The domain of the symbolic differentiation is represented as a network of BLUs, linked by pedagogical relationships. The main relationships used are *part-of*, *is-a*, *prerequisite*, *abstraction*, *opposite* and *similar*. *Part-of* and *is-a* relationships will provide us with hierarchical organisations suitable for using local search. Figure 13 shows a view of the domain using the part-of relationship.

Concerning the Instructional Action level, although its items are not represented in a hierarchical way in *Maisu*, a deep study shows that the IAs (see figure 14) can be organised in a taxonomy classified in two large classes: DAs and OAs.

The DAs are divided in two groups related to the IOs (knowledge and application[v]). For the *knowledge* IO, the tutor module includes several actions: explanation, example, counter-example and bibliographic reference, and for the *application* IO the actions are: exercise and test.

The OAs are also classified according to their main purpose: situating and motivating the learner during the session. The first class contains the OAs for introducing, finishing, summarising and informing; they are applied to several items of the session (*e.g.* concept, session, course, *etc*.) with different parameters (*e.g.* time from last session).

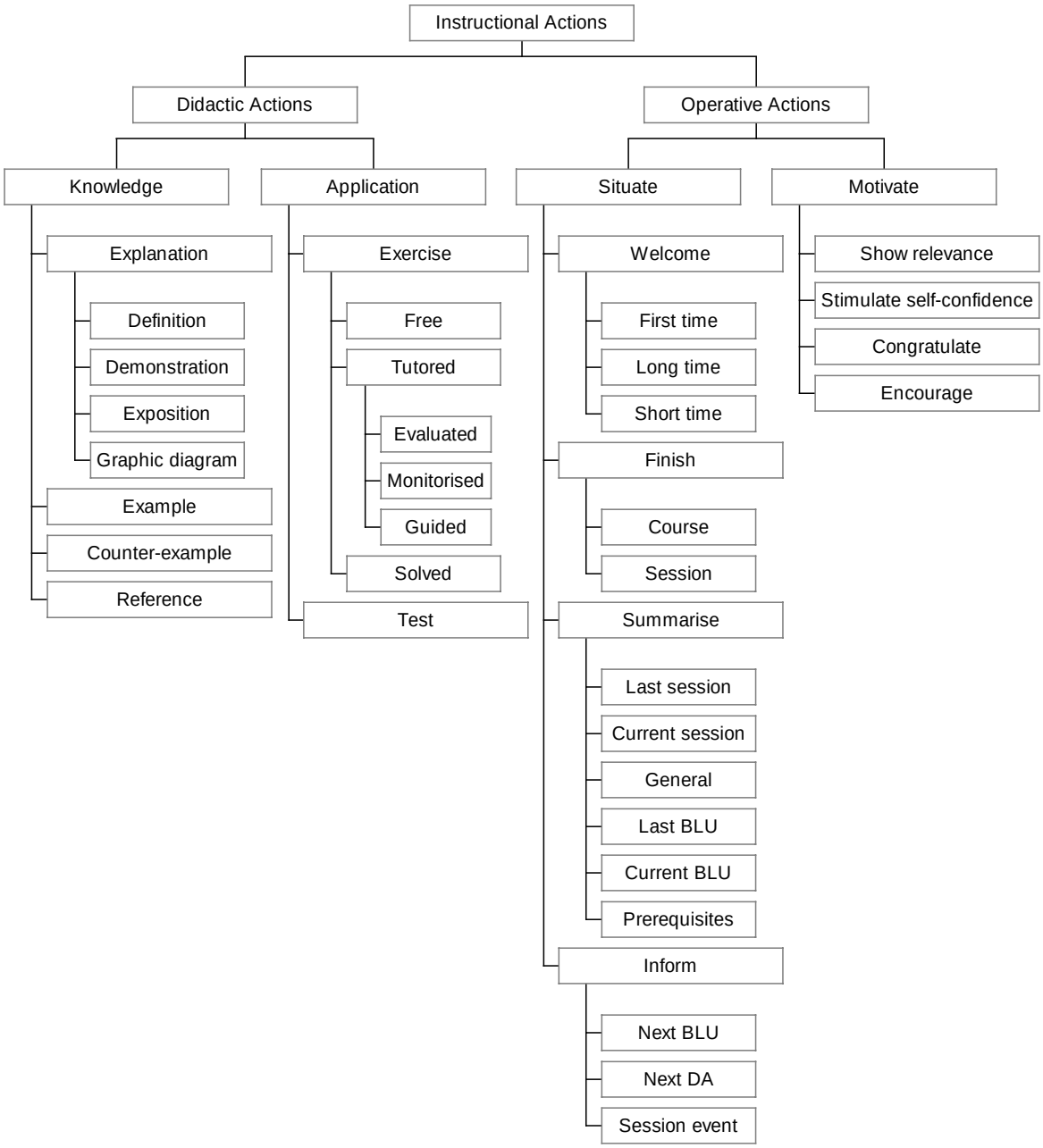

Figure 14. A hierarchical view of the Instructional Actions of *Maisu*.

Relevant Attributes in the planning process

The study of the features used by the original planner to build the plan particularly involves the original planner and the Student Model. In *Maisu*, the original planner is implemented as a production system, so the relevant attributes of the Student Model are obtained by analysing the condition elements of the rules. The Student Model of *Maisu* contains several types of information: the learner's general information such as name, identifier of the student group and background; student's preferences and learning style; and finally the levels of acquisition of each BLU.

Table 5. Student Model of *Maisu.*

Attributes of the student model	
General information	
Student-name:	<string>
Student-group:	<string>; *group of the student*
Background:	<string>
Time-from-last-session	(short, medium, long)
Session-number	(first, intermediate, last)
Preferences	
Preferred-duration:	[15..90]; *preferred duration of the session*
Intrusion-level	(low, medium, high)
Guiding-information-level	(low, medium, high)
Learning-preferred-style	(learning-by-examples, learning-by-doing, explanation-based learning, analogy-based learning)
Summarising-level	(low, medium, high)
Learning style	
Student-type	(novice, medium, expert)
Learning-speed	(low, medium, high)
Self-confidence-level	(low, medium, high)
Knowledge	
For all the BLU	
Level (BLU_i)	(zero, low, medium, high)

The attributes of the Student Model shown in table 5 compose the information that *Maisu* uses to plan. This information, properly represented in the CMK, will be used by the CBIP to link the objects of the IPM, to calculate the degree of matching between instructional situations, to search the IPM and to evaluate the results.

First of all, after analysing the types of the allowed values for the attributes, some different types arise. For example, the attributes representing the level of the student knowledge (`Level (BLUi)`) allow discrete values (`zero, low, medium, high`); the student's preferred duration of the session need a numeric value [`15 .. 90`]; and the student's learning-preferred-style allows exclusive values (learning-by-examples, learning-by-doing, explanation-based learning, analogy-based learning).

The most important features that must be taken into account in planning are those representing the knowledge of the student. Thus, these are essential to retrieve appropriate cases, can be used as indices and to evaluate the results of the learning session. In addition, the students can give their own opinion about their learning level.

Then, once the features are examined globally, they must be studied in relation to each level of the instructional plan, as each level takes into account a different set of attributes. In the example and that which concerns the BLUs level, the most relevant attributes are those related to the knowledge of the student, while in the IAs level, the features connected with the preferences and learning style of the student are more pertinent. Table 6 shows the data gathered during the analysis; it is focussed on the attributes relevant to the instructional planning task; for each attribute it indicates the levels of the plan and the tasks of the CBIP in which the attribute is used together with the corresponding weights (defined by the knowledge engineer).

Table 6. Results of the analysis phase.

Attribute	BLU	IA	Retrieve	Revise	Retain	Learner	Ret. Weight	Rev. Weight
Time-from-last session	●	●	●		●		0,5	
Session-number		●	●		●		0,5	
Intrusion-level		●	●		●		0,5	
Guidance-level		●	●		●		0,5	
Summarisation-level		●	●		●		0,5	
Preferred-learning-style		●	●		●		0,5	
Learner-type	●	●	●		●		0,5	
Learning-speed	●	●	●		●		0,5	
Self-confidence-level		●	●		●		0,5	
Level (BLU_i)	●		●	●	●	●	1	1
Selected-BLU-level		●	●		●		1	1
Duration	●	●	●		●		0,5	
Selected-BLU		●	●		●		0,5	
Selected-BLU-Difficulty		●	●		●		0,5	
Selected-BLU-type		●	●		●		0,5	

Step2: Adapting CBIP

The results obtained in the analysis of the concrete ITS are used to tailor the CBIP framework. The CBIP framework has been designed to lighten the construction of the CBIP. In this way, the work is localised mainly in the representation of the information gathered in the first step, in order to form the CMK structure. This meta-knowledge structure is organised around a compound object **`IP-structure`** (see figure 11). It contains objects representing the levels of the instructional plan, the items of each level, the attributes relevant at each level and the data types detected in the ITS. The framework includes a set of functions to create and relate these objects. The CBIP framework together with the implemented CMK already constitutes a simple planner.

The framework does not support extensively the phase of case reuse due to its knowledge-intensive nature. However, it provides an implementation of local search strategy and also a set of actions to adapt the instructional plan. The approach to the reuse phase that we recommend is the critic-based adaptation, due to its flexibility and simplicity of implementation by means of a set of rules. After observing the behaviour of the CBIP, the knowledge engineer should extract the rules that can improve the plans by performing slight changes. If s/he wants to use the local search strategy, s/he must include in the CMK the hierarchy of the plan items. This hierarchy is used in the strategy to find alternative items of a plan.

The CBIP framework can be more deeply customised by the knowledge engineer. Other customisation opportunity is provided in the retrieval phase. As it is said above, the framework includes a set of AI search algorithms that can even be augmented. The retrieval phase is guided by a Retrieval Strategy that establishes the algorithms used in the search, together with its parameters. The framework provides a basic strategy, but the knowledge engineer can modify it and experiment with the concrete planner to get an optimal one.

Although the framework is prepared to work with two levels, it can be adapted to manage more levels or just a one level plan. As it is shown in previous stages of this research (Elorriaga, *et al.*, 1995) a multileveled Case-Based Instructional Planner is feasible, hence the framework with a greater adaptation can be also the base for such a planner.

The Example: Adapting the CBIP to the target ITS

Table 7. Fragment of the CMK.

```
[IP-structure] of IP-STRUCTURE
  (level-num            2)
  (level-list           [BLU-level] [IA-level])
  (comment              "Structure of the Instructional Plan")
[BLU-level] of LEVEL-STRUCTURE
  (plan-item-type       BLU)
  (retrieve-att-list    [BLU-1-level] … [BLU-n-level] … [preferred-
duration] …)
  (revise-att-list      [BLU-1-level] … [BLU-n-level])
  (retain-att-list      [BLU-1-level] … [BLU-n-level])
  (learner-att-list     [BLU-1-level] … [BLU-n-level])
  (plan-item-list       [BLU-1] … [BLU-n])
  (hierarchy-available  TRUE)
  (plan-item-hierarchy  [HN-1-BLU])
  (comment              "Structure of BLU level")
[BLU-1-level] of ATTRIBUTE
  (value-type           [BLU-acquisition-level])
  (default-value        ZERO)
  (retrieve-app         TRUE)
  (revise-app           TRUE)
  (retain-app           TRUE)
  (learner-app          TRUE)
  (retrieve-weight      1)
  (revise-weight        1)
  (revise-change        TRUE)
  (learner-question     "How do you think is your knowledge about
                        derivatives (Zero / Low / Medium / High)?")
  (comment              "Attribute: level of BLU-1 derivatives")
[BLU-acquisition-level] of VALUE-TYPE
  (type                 discrete-values)
  (value-list           ZERO LOW MEDIUM HIGH)
[HN-1-BLU] of HIERARCHY-NODE
  (plan-item            [BLU-1])
  (ancestor             NIL)
  (descendant-list      [HN-2-BLU] [HN-3-BLU])
  (replaceable          FALSE)
  (alternative          FALSE)
[BLU-1]of PLAN-ITEM
  id                    DERIVATIVES
…
```

In the second step the information already gathered is encoded in the CMK Module. Table 7 shows a subset of the objects that compose the CMK for *Maisu*. The instructional plan produced by the CBIP is composed of two levels `BLU-level` and `IA-level`. In the first level, the relevant attributes are mainly those that represent the knowledge of the student (`BLU-i-level`, i = 1 .. n). They are used in the retrieval (`retrieve-att-list`) and the revision of cases (`revise-att-list`), the organisation of the IPM (`retain-att-list`) and also the system can ask the student about them (`learner-att-list`). We point out that different sets of attributes can be taken into account in each of these four aspects. `BLU-1-level` is an attribute example that represents the knowledge of the student about `BLU-1` (derivatives). This attribute is appropriate for the four issues i.e. `retrieve-app, revise-app, retain-app, learner-app`, and, therefore, the weights to apply when retrieving (`retrieve-weight`) and

revising (`revise-weight`) are included as well as the question to be asked to the learner (`learner-question`). The type of values allowed for this attribute (`BLU-acquisition-level`) is a discrete set. In addition, for the level of BLU, there is a plan item hierarchy available based on the graph shown in figure 13 whose root item is (`HN-1-BLU`).

Once the CMK is completed, the planner should be enhanced with adaptation critics. The critics use ITS-specific knowledge to perform some modifications on the instructional plans. The knowledge engineer should construct critics that detect and solve several problems at each level of the plan. For example, instructional plans with an inappropriate duration, repeated items in one level of the instructional plan, non-optimal order of the items within the plan, *etc*.

Table 8 shows a sample critic of the BLU level. It checks if any of the BLUs (*?blu*) included in the instructional plan have a prerequisite (*?blu-pre*) that is not already reached by the learner nor included in the plan. If this situation happens, *?blu-pre* is included in the plan ahead of *?blu*. This example illustrates how the critic can use knowledge of the concrete ITS (prerequisite relationship) to perform a more informed adaptation. The critics can use the local search to find alternative items of the instructional plan because the CMK also incorporates the hierarchy of plan items.

Table 8. Example of a critic.

```
CRITIC C-BLU-PRE-1

(defrule c-blu-pre-1
  (declare (salience 900))
  ?plan <- (object (name [PS-BLU]) (level blu) (item-list $?il))
  ?blu <- (object (is-a BLU) (prerequisites $?blu-pre-l))
  (test (member$ ?blu ?il))
  ?blu-pre <- (object (is-a BLU))
  (test (member$ ?blu-pre ?blu-pre-l))
  (test (and (not (reached-p ?blu-pre))
             (not (member$ ?blu-pre ?plan))))
=>
  (insert-item ?blu-pre (member$ ?blu ?il) ?plan)
  (bind ?*adap-degree* (+ ?*adap-degree* 0.2))
)
```

In addition, the retrieval strategy can also be tailored by the knowledge engineer. For example, the table 9 presents a possible configuration. This strategy sets that the IPM is first explored with a heuristic algorithm that uses the similarity function SF (see figure 5) both as a heuristic function as well as to determine the degree of matching; the matching threshold is 0.8 and it returns at the most one case. When this algorithm does not succeed, an exhaustive one is called with a lower threshold, (0.7).

Table 9. Example of retrieval strategy.

```
Retrieval Strategy
Case-list := Run (Hill-climbing-algorithm, SF, SF, 0.8, 1)
If empty (case-list) Then
      Case-list := Run (exhaustive-algorithm, SF, 0.7, 1)
```

Step 3: Integrating CBIP

Once the CBIP is tailored to a concrete ITS, the next step is to integrate the new planner inside the tutoring system. The enhanced ITS will be able to use both planners, hence the Didactic Instructor must be modified to be able to select the most adequate plan between both planners. It is extremely complex to compare plans (Pollack, 1995) because of the difficulty to specify the appropriate metrics. Thus, the solution proposed is based on the information available from the CBIP, *i.e.* the created plan estimate calculated by the Assessment Component. Unfortunately, this value can not be obtained for the plan built by the original planner. Therefore, the solution we apply to handle this situation is to first call the CBIP and if the estimate of the plan it builds

does not exceed a pre-established threshold then call the original instructional planner to apply its plan.

The integration of the CBIP, implemented in CLIPS, into an ITS is straightforward assuming that both systems are implemented with the same programming language. Some difficulties arise when the implementation languages are different. However, the ease to combine CLIPS with C code and C with other programming languages makes this problem manageable. In this case, C will be used as a bridge to communicate the CBIP to other programming languages.

Step 4: testing

In the testing phase, two types of tests must be fulfilled. From a computational point of view, the AI programmer must check the correctness of the adaptation and integration of the CBIP. On the other hand, the performance of the HSIIP should be tested by the knowledge engineer in order to check whether the important features have been taken into account and to verify the correctness of the parameters (*i.e.* the thresholds) used through the planning process. This process can be carried out by means of a refinement cycle within which different thresholds are experimented.

RESULTS

This section presents the experiments carried out to evaluate the HSIIP approach together with their resulting data and our conclusions. The objective of these experiments is to evaluate the performance of the HSIIP in terms of the changes in the students' knowledge.

Description of the experiments

The experiments design includes a set of classical instructional planners, their correspondent HSIIP and a population of simulated students. In addition, the testing is based on two premises justified on the difficulty of testing isolated modules in an ITS (Mark & Greer, 1993): (1) In order to minimize the influence of the other ITS modules, the experimentation prototype is composed only of the instructional planner. (2) Besides, in order to perform a significant number of experiments under the same initial conditions the planners are tested with simulated students. They make possible to probe the instructional planner isolated due to their computational nature, so they have been proposed as useful artefacts for formative evaluation (VanLehn et al., 1994).

The experiment simulates the development of the same instructional session for several students and the results are calculated in terms of the change of the acquisition level for each session topic.

The simulated students are classified in four groups according to their learning characteristics. The learning process of the students has been simulated on the basis of a table that relates plans and student groups with a range of possible learning results which represents the knowledge acquired by the student after finishing the session. The final result is chosen randomly from this range.

Four versions of a simple classical instructional planner are used (called CIP-*x*, *x*=1,..,4). CIP-1 is the simplest planner missing 20% of *Maisu's* rules, while CIP-2 misses 10%. CIP-3 is the original planner of *Maisu*. Finally, CIP-4 is a hypothetical planner that uses the table of results of the plans for each student group to get the optimal plan. Therefore, each CIP includes different planning strategies with different outcome. The performance of the classical planners (see table 10) is measured taking into account the mean value of the effect of its plans on the simulated students' knowledge level. For example, the simulated students learn on average a 30% of the lesson with CIP-1. For each classical planner, a version extended with the HSIIP approach has been built (called HSIIP- *x*, *x*=1,..,4). The adaptation critics of these HSIIPs are quite simple and equal in all the versions. They control the duration of the session and the level of achievement of the BLUs.

Table 10. Performance of the classical instructional planners.

Version of the planner	Performance
CIP-1	30%
CIP-2	50%
CIP-3	70%
CIP-4	100%

Each version of the CIP and its corresponding HSIIP is tested with the set of simulated students uniformly distributed among the four groups. The experiment includes a number of stages and in each stage plans for a group of 400 students are generated (100 of each group). The HSIIP starts with an empty IPM. Therefore, in the first stage only the CIP can create the plans (training phase) and the CBIP incorporates them gradually in the IPM. The next ten stages correspond to the collaboration phase in which HSIIP works using both CIP and CBIP. It can be observed that the experiment gets meaningful results in eleven stages (see figure 15). In the collaboration phase, the estimate of the plans built by the CBIP must exceed the acceptance threshold (0.8), otherwise they will be requited of the CIP.

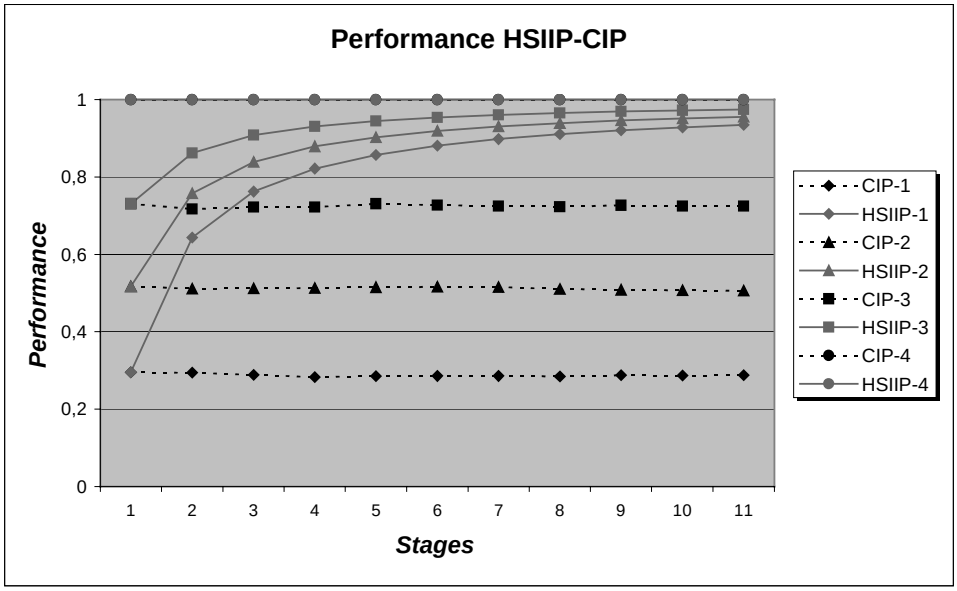

Figure 15. Performance of CIP and HSIIP.

Analysis of the results

The results of the experiments are shown in the graphs of figure 15 and 16. The main feature evaluated is the result of the plans generated by each planner calculated in terms of the learning level of the session topics that the students get.

Figure 15 shows the evolution of the mean value of the results of the plans generated by the CIPs and by the corresponding HSIIPs. Obviously, the CIPs maintain their performance unchanged along the experiment and equal to the values presented in table 10. On the other hand, it is remarkable that the HSIIPs improve their performance asymptotically reaching the best outcome. In the first stage, each HSIIP matches the performance of its corresponding CIP, in fact, in this stage the CIP is working alone. In the seventh stage, i.e. after planning for 2800 students (7^{th} stage) the performance of all the HSIIPs exceed 0.9. In addition, all the HSIIP

versions of the planner get a similar performance in the last stages. However, the HSIIP with the worst CIP need more training to get those results, but obtains a greater improvement.

Figure 16 shows two related graphs that correspond to HSIIP-2, the upper one represents the evolution of the use of the CIP-2 and the CBIP-2 during the experiment. It can be observed that after the training stage the CBIP-2 starts planning very quickly, and after the second stage it is used almost exclusively. The lower graph shows the result of HSIIP-2 in the same scale. It can be verified that the more the CBIP-2 is used, the better the result of the HSIIP-2.

Performing an expert analysis of the state of the IPM (Mark & Greer, 1993), it can be observed that the cases that represent the application of plans in inappropriate situations are seldom used, so these conditions are quickly detected. On the other hand, the cases that represent the best applications are extensively reused.

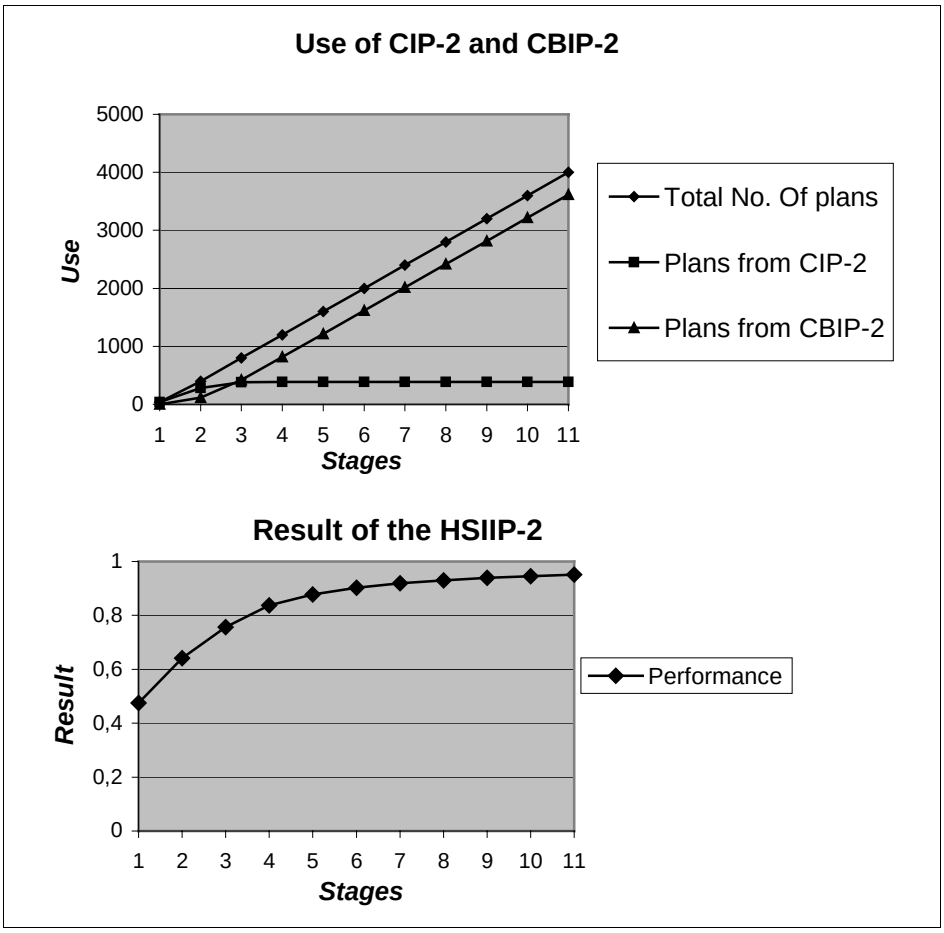

Figure 16. Evolution of the result of HSIIP-2 contrasted with the use of the CIP and the CBIP.

Even though the evolution of the HSIIP is similar in all the versions, the original CIP sets the knowledge the CBIP receives in the training phase. Therefore, if the CIP does not know that a plan is good for a situation, neither it nor the CBIP will use it. This behaviour can be modified including some experimentation rules when selecting and adapting the plans.

RELATION TO OTHER SYSTEMS

Instructional planning has been an important task inside the ITSs. For some years, the debate on the necessity and appropriateness of having an instructional plan guiding the learning sessions has been active. The constructivists blame the ITSs for planning a narrow teaching interaction, controlled by the system. They promote, in contrast, more learner-centred interactive learning systems. The most rigid Constructivism assumes that the computer is nothing more than a passive medium, in which the student works and constructs his/her knowledge. This interesting

theme was discussed in (Brna *et al.*, 1996) and some authors (Vassileva & Wasson, Wasson, 1996) agreed in defending the utility of planning the instruction but away from a rigid instructivist point of view. Vassileva & Wasson (Vassileva & Wasson, 1996) conclude that Instructional Planning can be used in different teaching styles such as tutoring, coaching, cognitive apprenticeship, Socratic dialogue and combinations of these to provide consistency, coherence and continuity to the learning process. Furthermore, (Wasson, 1996) suggests specific applications of Instructional Planning: to balance inductive and deductive learning and activities, promote learners in detecting their own errors, make the goal of learning clear, planning collaborative lessons, determining when to decrease scaffolding and managing the learning session. Some authors (Luckin, 1996; Arruarte *et al.*, 1996) try to combine both constructivism and instructivism in a pragmatic way to produce ITSs that, guided by a non-rigid instructional plan, offer more help to the student in constructing his/her own knowledge.

Summarising all those opinions, it can be concluded that a flexible Instructional Planning is useful for several tasks inside the ITSs in order to provide a more adaptable medium of learning (McCalla, 1992) and this can be compatible even with constructivism.

Different approaches have been used to tackle the Instructional Planning task (reviewed in (Macmillan & Sleeman, 1987) (McCalla, 1992) (Vassileva, 1995), from human-authored plans to flexible AI-based instructional planners, trying, in all cases, to construct more adaptable ITSs (McCalla, 1992). Within the wide spectrum of approaches and techniques, Case-Based Reasoning (CBR) has been little used in the Instructional Planning Task. However, some interesting work has explored this possibility. CBR was proposed as a method for instructional plan building in (Riesbeck & Schank, 1991) by means of Reactive Action Packages (RAPs). A RAPs, in the teaching context, is a small set of instructional actions responsible for achieving certain instructional goals. This approach is close to rule-based planners but representing the RAPs as memory structures.

With a different aim, the Science Education Advisor (SCI-ED) is created (Kolodner, 1991; Kolodner 1993), a system for aiding elementary school teachers with the creation of lessons in science.

At the same time, other researchers (Du & McCalla, 1991) presented a Case-Based Mathematics Instructional Planner. This project consisted of translating the Case-Based Planning Approach (Hammond, 1989) to the instructional area, with two objectives: producing plans for the teachers and for ITSs. This Case-Based Planner does not take into account any feedback from the execution of the plan.

A more recent work (Khan & Yip, 1995-b) presents a Case-Based Task Manager for Case-Based Teaching Systems that evaluates the constructed plans identifying successful and failing plans.

These works demonstrate that CBR is also a valid paradigm for the Instructional Planning Task with some advantages over first principles planning (see section 2). In particular, the inherent ability to learn of the Case-Based Systems must be emphasised. However, in the systems introduced above, the learning feature is not deeply exploited; one of them (Du & McCalla, 1991) stores the created plans, while the system by (Khan & Yip, 1995-b) performs a simple and manual evaluation of the result of the case before storing it. Nevertheless, the learning ability should not be restricted to the storing of the new experiences. This basic learning should be combined with statistical learning strategies that count the different aspects of the results that the instructional plan obtains, as it has been done in the HSIIP. This information will improve the Case-Based Instructional Planning by aiding in the retrieval of cases and in the assessment of the instructional plans built.

Other authors (Macmillan & Sleeman, 1987; Macmillan, *et al.*, 1988) presented a Self-Improving Instructional Planner (SIIP) over a blackboard architecture. SIIP co-operates with a module called Plan Improvement Experimenter (PIE) that receives an improvement goal and responds with a plan for it. PIE monitors the result of applying the plans to decide whether to accept or reject the changes and also to record unanticipated outcomes. Macmillan and Sleeman pointed out the representation of domain-specific knowledge and the possibility of applying the planner to different domains as future work. They claimed that the SIIP could be applied to other domains using knowledge acquisition techniques in order to incorporate the domain-specific knowledge and afterwards use the PIE for tuning the planner. From this work, it can be concluded the necessity of an effort to construct a domain-independent instructional planner that

can be adapted to concrete domains, besides the usefulness of experimental learning inside a planner. This issue can be handled using the CBR paradigm separating the domain-dependent knowledge and representing it explicitly, as the framework of CBIP does. In addition, the learning abilities of the CBIP allow the experimentation and tuning of the planning knowledge.

The hybrid approach of HSIIP is similar to the PRODIGY/ANALOGY system for planning (Veloso, 1994). In this system, a case-based planner is combined with a basic planner based on a search algorithm. The former maintains the information of previous executions of the later; this information is used to speed up the search in future planning tasks by pruning branches that failed in past experiences.

CONCLUSIONS

In this paper we have presented a new approach to develop self-improving ITSs and a general methodology to apply it. This approach is supported by the learning capability of the case-based reasoners; its core is a Case-Based Instructional Planner (CBIP) that learns by storing the results of its previous experiences.

The design of the Hybrid Self-Improving Instructional Planner (HSIIP) lies in the possibility of integrating the CBIP into an existing ITS provided with a more traditional planner, in order to endow it with the learning ability. Initially, the CBIP will learn by observing the behaviour of the original instructional planner and recording the teaching experiences in the Instructional Plan Memory (IPM). When enough cases have been recorded in the IPM, CBIP will produce instructional plans by itself. If the CBIP adapts, to a great extent, the cases used to generate the plan, they will also be inserted in the IPM.

The behaviour of this hybrid instructional planner is sounder than the behaviour of the CBIP on its own, because both planners support it. When the CBIP is not able to generate a suitable plan, the tutor can use the original planner. On the other hand, The CBIP does not need a manual initialisation of the Case Memory with a basic case set, this task is done automatically benefiting from the pedagogic planning knowledge included in the original instructional planner.

The learning of the HSIIP is not limited to record cases, CBIP gathers the results of the application of the plans to estimate the degree of appropriateness of future plans. So, the planner gives priority to the cases with best results while those plans with bad results are not reused. The CBIP uses heuristic formulae that take into account the system's and the learner's points of view. The outcome of this statistical learning is separated in a computational dimension, in which the number of times that a case is used is noted down, and an educational dimension in which the beliefs of both the system and the learner are also recorded.

The HSIIP has been successfully evaluated using simulated students. The hybrid planners learn the result of the plans and perform better than the original ones. This means that the HSIIP gathers the plans from the original planner and learns from its experience what are the best cases to apply in each instructional situation.

Taking into consideration the case-based instructional planning separately, the use of CBR (Case-Based Reasoning) to develop instructional planners for new ITSs can solve some of the problems of developing such time-consuming knowledge-based systems. In general, an instructional planner built on the basis of CBR does not require an explicit model of tutoring, the knowledge elicitation becomes a task of gathering cases (Watson & Marir, 1994). The implementation of a case-based instructional planner is reduced to identifying significant features that describe the instructional sessions and foster it with an initial set of cases. Finally, the maintenance of the case-based instructional planner is an easier task because it is prepared to acquire more knowledge by means of new cases.

Concretely the work presented here also focuses on the idea of reducing the effort in producing case-based instructional planners. With this aim an object-oriented framework has been designed and implemented in CLIPS; it embodies the common objects and methods for developing this kind of planners. In order to facilitate the reusability, the framework separates explicitly the general functionalities from the specific knowledge of the ITS that is used in the process of instructional planning, so it can be adapted to several domains. In addition, it provides the knowledge engineer with the opportunity of tailoring and enhancing the process of

instructional planning. This framework can be applied for improving existing ITSs within the HSIIP approach, as well as for creating planners for new ITSs. Finally, we have defined a methodology for developing HSIIPs supplemented with a set of guidelines to fulfil each task. The framework and the methodology have been designed in such a way that the original ITS needs only slight changes.

More details of this project can be found in (Elorriaga, 1998). This work is still open and can be completed and improved in different ways. A very interesting issue is the case adaptation as a knowledgeable transformation will produce better CBIP's performances. A deep study of the different opportunities to reuse the cases will be necessary in order to identify general knowledge useful in the adaptation process. The measurement of the transformation performed while adapting has been also treated in a shallow way and so it could become an interesting research line. In addition, the framework can be enhanced with a friendly interface to help the development of CBIPs following the methodology here presented. Finally, it would be very interesting to explore the possibility of introducing a degree of experimentation in the generation of the plans by the hybrid planner in order to test different plans.

Acknowledgements

The authors want to thank the anonymous reviewers and John Self for their help and useful comments on the paper. Many people help us in different stages of this thesis project, among them, Jim Greer, Gordon McCalla, Helen Pain and all the members of the GTI research group at the University of the Basque Country, thank you very much! This work is partly supported by the Department of Education, Universities and Research of the Basque Country Government-*Eusko Jaurlaritza* and the University of the Basque Country-*Euskal Herriko Unibertsitatea* (UPV 141.226-TA109/99).

References

Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7, 39-59.

Arruarte, A., Fernández-Castro, I. & Greer, J. (1996). The CLAI Model: A Cognitive Theory of Instruction to Guide ITS Development. *Journal of Artificial Intelligence in Education*, vol. 7 (3/4), pp. 277-313.

Arruarte, A., Fernández-Castro, I., Ferrero, B. & Greer, J. (1997). The IRIS Shell: "How to Build ITSs from Pedagogical and Design Prerequisites. *International Journal of Artificial Intelligence in Education*, vol. 8 (3/4), pp 341 381.

Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H. & Krathwohl, D.R. (1956). Taxonomy of Educational Objectives: Handbook I, Cognitive Domain, Longman.

Brna, P., Paiva, A. & Self, J (1996). *Proceedings of European Conference on Artificial Intelligence in Education* (Euro-AIED'96).

Christodoulou, E. & Keravnou, E.T. (1997). Integrating multiple problem solvers in knowledge-based systems. *The Knowledge Engineer Review*, 12 (2), 181-207.

Cognitive Systems (1992).ReMind Developer's Reference Manual. Boston.

Coulman, R. (1991). Combining Case-Based Reasoning and Granularity for Educational Diagnosis in an Intelligent Tutoring System. Research report 91-9, ARIES laboratory, Department of Computational Science, University of Saskatchewan, Canada.

Dillenbourg, P. (1989). Designing a self-improving tutor: PROTO-TEG. *Instructional Science*, 18, 193-216.

Du, Z. & McCalla, G.I. (1991). A Case-Based Mathematics Instructional Planner. *Proceedings of the International Conference on the Learning Sciences*, Northwest Univ., 122-129.

Elorriaga, J.A., Fernández-Castro, I. & Gutiérrez, J.(1995). Case-Based Reasoning for Self-Improving Intelligent Tutoring Systems. Proc. Int. Conf. on Computers in Education (ICCE'95) Singapore, 259-266.

Elorriaga, J.A. & Fernández-Castro, I. (1996). The HSIIP Approach. An Extension of a Teacher's Apprentice. Proc. Int. Conf. on Intelligent Tutoring Systems (ITS'96) Montréal, Canada, LNCS, 401-410.

Elorriaga, J.A. & Fernández-Castro, I. (1997). Design of a Kernel for the Development of Case-Based Instructional Planners. Proc. Int. Conf. on Artificial Intelligence in Education (AI-ED'97) Kobe, Japan, IOS Press, 580-582.

Elorriaga, J.A. (1998). Planificación de la instrucción en Sistemas Tutores Inteligentes Evolutivos desde un enfoque de Razonamiento Basado en Casos. PhD Thesis. University of the Basque Country, Donostia.

Fernández-Castro, I., Díaz de Ilarraza A. & Verdejo, F. (1993). Architectural and Planning Issues in Intelligent Tutoring Systems. Journal of Artificial Intelligence in Education, 4 (4), 357-395.

Gutiérrez, J., Fernández-Castro, I., Díaz de Ilarraza, A. & Elorriaga, J.A. (1993). General Architecture and Blackboard Model for a Training Tutor. *Proceedings of the World Conference on Artificial Intelligence in Education*. (AI-ED'93). Edinburgh (Scotland), 34-41.

Gutiérrez, J. (1994). INTZA: un Sistema Tutor Inteligente para Entrenamiento en Entornos Industriales. PhD Thesis. University of the Basque Country, Donostia.

Gutstein, E. (1992). Using Expert Tutor Knowledge to Design a Self-Improving Intelligent Tutoring System. *Intelligent Tutoring Systems, 2nd Int. Conference ITS'92*, Frasson, C., Gauthier C., McCalla, G.I. (Eds.), LNCS Springer-Verlag, 625-632.

Hammond, K.J. (1989). Case-Based Planning, Chandrasekaran, B. (Ed.), Academic Press.

Kimball, R. (1982). A self improving tutor for symbolic integration. *Intelligent Tutoring Systems*, 283-307. Sleeman, D., Brown, J.S. (Eds.), Academic Press.

Khan, T. & Yip, Y. J. (1995-a) CBT II - Cased-Based computer-aided instruction: survey of principles, applications and issues. *The Knowledge Engineering Review*, 10 (3), 235-268.

Khan, T. & Yip, Y. J. (1995-b) Case-based task management for computer-aided learning. *Progress in Case-Based Reasoning*, Watson, I.D. (Ed.), 201-209.

Kolodner, J. L. (1991). Helping Teachers Teach Science Better: Case-Based Decision Aiding for Science Education. The International Conference on the Learning Sciences, Proceedings of the 1991 Conference, Birnbaum, L. (Ed.), AACE, 274-280.

Kolodner, J. L. (1993). Case-Based Reasoning. Morgan Kaufmann Pub.

Luckin, R. (1996). TRIVAR: Exploring the "Zone of Proximal Development". *Proceedings of European Conference on Artificial Intelligence in Education* (Euro-AIED'96). Brna, P., Paiva, A. & Self, J (Eds.), 16-22.

MacMillan, S.A. & Sleeman, D.H. (1987). An architecture for a self-improving instructional planner for intelligent tutoring systems. *Computational Intelligence*, 3, 17-27.

MacMillan, S.A., D. Emme & Berkowitz, M. (1988). Instructional Planners: Lessons Learned. *Intelligent Tutoring Systems: Lessons Learned*, Psotka, J., Massey, L.D. & Mutter, S.A. (Eds.), Lawrence Erlbaum Associates pub, 229-256.

Mántaras R. & Plaza E. (1997). Case-Based Reasoning: an overview. *AI Communications*, 10, 21-29.

Mark, M. and Greer, J.E. (1993), "Evaluation Methodologies for Intelligent Tutoring Systems", *Jl. of Artificial Intelligence in Education*, AACE, Charlottesville, 4(2/3), 129-153.

McCalla, G.I. (1992). The Search for Adaptability, Flexibility and Individualization: Approaches to Curriculum in Intelligent Tutoring Systems. *Adaptive Learning Environments*, Jones, M. & Winne, P.H. (Eds.), Springer-Verlag, 91-121.

Murray, W.R. (1990). A Blackboard-Based Dynamic Instructional Planner. Research Report R-6376, Artificial Intelligence Centre, FMC Corporation, Santa Clara. USA.

O'Shea, T. (1982). A self improving quadratic tutor. *Intelligent Tutoring Systems*, D. Sleeman & J.S. Brown (Eds.), Academic Press, 309-336.

Pollack, M. (1995). Evaluating Planners, Plans and Planning Agents. *SIGART Bulletin*, vol.6 (1), pp: 4-7.

Porter, B., Bareiss, R. y Holte R. (1990). Concept Learning and Heuristic Classification in Weak Theory Domains. In *Artificial Intelligence*, 45 (1/2), 229-263.

Riesbeck, C.K. & Schank, R.C. (1989). Inside Case-Based Reasoning. Hillsdale, NJ: Lawrence Erlbaum Associates.

Riesbeck, C.K. & Schank, R.C. (1991). From Training to Teaching: Techniques for Case-Based ITS. *Intelligent Tutoring Systems: Evolutions in Design*, Burns, H., Parlett, J.W. & Redfield, C.L. (Eds.), Lawrence Erlbaum Associates Pub, 177-193.

Rumbaugh, J. Blaha, M., Premerly, W., Eddy, F. & Lorensen, W. (1991). Object-Oriented Modeling and Design. Prentice Hall, Inc.

Schank, R.C. (1982). Dynamic Memory: a Theory of Reminding and Learning in Computers and People. Cambridge University Press.

VanLehn, K. Ohlsson, S. and Nason, R. (1994). "Applications of Simulated Students: An Exploration", Jl. of Artificial Intelligence in Education, AACE, Charlottesville, 5(2), 135-175.

Vassileva, J. & Wasson, B. (1996). Instructional Planning Approaches: form Tutoring towards free Learning. *Proceedings of European Conference on Artificial Intelligence in Education* (Euro-AIED'96). Brna, P., Paiva, A. & Self, J (Eds.), 1-8.

Vassileva, J. (1995). Reactive Instructional Planning to Support Interacting Teaching Strategies. *Proceedings of World Conference on Artificial Intelligence in Education* (AIED'95). Greer, J. (Ed.), 334-342.

Veloso, M.M. (1994). Planning and Learning by Analogical Reasoning. LNAI, Springer-Verlag.

Wasson, B. (1990). Determining the Focus of Instruction: Content Planning for Intelligent Tutoring Systems. PhD Thesis. University of Saskatchewan. Canada.

Wasson,B. (1996). Instructional Planning and Contemporary Theories of Learning: Is it a Self-Contradiction?. *Proceedings of European Conference on Artificial Intelligence in Education* (Euro-AIED'96). Brna, P., Paiva, A. & Self, J (Eds.), 23-30.

Watson, I. & Marir, F. (1994). Case-Based Reasoning: A Review. The Knowledge Engineering Review, 9, 327-354.

Notes

[i] The student's level of acquisition of `topic` has changed from `low` to `medium` during the session.

[ii] This is a short way to refer to the factors used in the following formulae

[iii] The local search substitution (Kolodner, 1993) method tries to find alternative items to interchange with items of the plan that are not appropriate. It needs an abstraction hierarchy to search an appropriate candidate to substitute for the original one.

[iv] The diagram is based on the *Object Modelling Technique* (OMT) (Rumbaugh *et al*., 1991).

[v] The IOs used in this tutor are knowledge and application (taken from the Bloom's taxonomy (Bloom, 1956)) applied to the corresponding concepts of the domain.