Journal of Universal Computer Science, vol. 11, no. 9 (2005), 1530-1542
submitted: 15/2/05, accepted: 30/6/05, appeared: 28/9/05 © *J.UCS*

Authoring and Diagnosis of Learning Activities with the *KADDET* Environment

Begoña Ferrero
(University of the Basque Country, Spain
bego.ferrero@ehu.es)

Maite Martín
(University of the Basque Country, Spain
jibmarom@si.ehu.es)

Ainhoa Alvarez
(University of the Basque Country, Spain
ainhoa.alvarez@ehu.es)

Maite Urretavizcaya
(University of the Basque Country, Spain
maite.urretavizcaya@ehu.es)

Isabel Fernández-Castro
(University of the Basque Country, Spain
isabel.fernandez@ehu.es)

Abstract: This paper describes *KADDET*, a cognitive diagnostic environment created to assess the conceptual and procedural learning activities of students. It is composed of a diagnostic engine, DETECTive, and a knowledge acquisition tool developed to fulfil its knowledge representation needs, KADI. Both of them share a Model of Learning Tasks (MLT) as a diagnostic basis. One of the main goals of this environment is to provide teachers with easy-to-use tools that facilitate the construction of learning environments with diagnosis capabilities customized to their particular subject domains and adaptation styles.

Keywords: Learning Environments, Cognitive diagnosis, Authoring Tools, Error libraries, Model Tracing
Categories: K.3, K.3.1

1 Introduction

The latest developments in educational computer sciences have made reference to several approaches to educational systems, but regardless of the learning method used, what they all share is *the student's learning activities*. These activities play a major role because they encourage "*learning by doing"* [Anzai & Simon 79], reinforce the knowledge acquired and can even be used from a perspective of self-assessment. In addition, the results obtained can be used to identify conditions for adapting the current teaching/learning strategies. In this context, the *level of user adaptation* is seen as a crucial issue for improving the learning process. This aspect

has usually been tackled by including specific components devoted to diagnostic functions in educational systems. However, the diagnosis capability involves a development that is not trivial, which means that it can be more or less simple depending on the type of learning activity, but that it can become quite complex when working with procedural domains.

On the other hand, building adaptive systems from scratch has proved to be so difficult that it prevents their massive use [Murray 97, 03]. This has resulted in the creation of authoring tools aimed at enabling teachers to build their adaptive learning systems tailored to the selected domains.

In this paper we present the *KADDET* environment, which focuses on generating adaptive systems oriented to the performance and diagnosis of learning activities. We have defined a hybrid and generalised diagnostic approach that combines several techniques integrated within a Model of Learning Tasks (MLT). *KADDET* is comprised of two main systems: DETECTive, a diagnostic engine capable of evaluating learning activities related to any domain—provided it is well described according to the requeriments specified by the MLT; and KADI, as a complementary authoring tool aid oriented to teachers. In the next section we present the main objectives of the proposal and some related work. Then, the general characteristics and structure of the above mentioned systems are described. Finally, we will draw some conclusions and suggest future lines for research.

2 Main Objectives and Related Works

This work has pursued a double goal: first, to define and implement a generic diagnostic engine customizable and valid for a wide range of domains, and second, to build an appropriate authoring tool usable by the teachers. Thus, during the acquisition phase, the authoring tool automatically customizes the diagnostic engine to the specific domain according to the teacher's requirements. Later on, the student uses the tailored system to perform learning activities that will be diagnosed according to the previously supplied domain description—diagnosis phase. This approach provides a supporting tool with monitoring and diagnosing capabilities for learning, which should be complemented by other means, i.e. by a conventional educational process or through an external learning system.

Various approaches to the diagnosis of learning tasks have achieved interesting results. However, most of them need a hard knowledge of engineering work that the authoring process is supposed to help overcome. In the next sections we present the most promising diagnosing techniques, a study of authoring tools from a generic diagnosis perspective and, finally, our starting hypothesis.

2.1 Techniques for Cognitive Diagnosis

The cognitive diagnosis field mentions several diagnostic techniques. Among them, Error libraries, Model-Tracing and Constraint Based Modelling are the most widely used approaches; they have been applied to several types of domains and are claimed to be generic. Other techniques—such as Machine Learning [Kono *et al.* 94], Bayesian Belief Networks [Millán *et al.* 00] and Fuzzy Logic [Katz *et al.* 94]— have also yielded promising results for students' diagnosis. However, most of them require

a deep knowledge representation that closely relates them to the domain and, therefore makes them unfeasible for a generic system.

Error libraries [Burton 82] are based on the explicit representation of erroneous knowledge obtained from the recording and interpreting of wrong answers given by students. Although its interest is evident, the cost of building such libraries is very high [Baffes *et al.* 96], and not very realistic if the teacher is solely responsible for it.

The *Model-Tracing* technique [Anderson *et al.* 90] is easy to implement as it does not require exhaustive studies or complex techniques, and has a low computational cost. It consists of a step-by-step monitoring of the student's actions with regard to one or more problem solving models. The differences found among the correct and tentative solutions reflect the learner's deviations. This technique requires a set of solution models whose completeness determines diagnosis reliability [Ohlsson 94].

Constraint-based modelling [Mitrovic *et al.* 99] expresses the domain as a set of constraints on correct solution paths. It does not require a runnable expert module, a bug library or a sophisticated inference mechanism. Nevertheless, the estimated cost of representing and verifying the domain model as a set of constraints is rather high [Ohlsson 94] [Suraweera & Mitrovic 04].

Although the techniques considered are claimed as generic, taken separately, none of them fully meets an environment's needs for building diagnostic systems. For instance, they would need a *complete set* of problem solving models, constraints or error libraries and might not be valid for different types of domain. So, our approach tries to alleviate problems of completeness and domain customization by adequately combining a group of techniques. Hybrid diagnosis approaches, mainly centered on plan recognition, have already been successfully used in tutoring systems [Greer & Koehn 95][Goldman *et al.* 99], showing that the combination of techniques yields better results than each individual technique by itself.

2.2 Diagnosis Issues in Authoring Tools for Learning Systems

Authoring tools for teaching/learning systems are many and diverse in both goals and characteristics [Murray *et al.* 03]. For the purposes of this study, we will only consider those that build practice-oriented systems, i.e. systems that provide students with environments enabling them to put their knowledge into practice, and give the advice required to "learn by doing". SIMQUEST, RIDES, XAIDA and Demonstr8 all belong to this group of systems.

SIMQUEST [Joolingen *et al.* 96] focuses on the conceptual characteristics of the domain and allows discovery learning environments using simulations to be created. The diagnosis is made only by comparing the learner's final result with the result expected. The XAIDA system [Hsieh *et al.* 99] is suitable only for learning perfectly identified maintenance tasks, and applies the *model tracing* paradigm to monitor the students' activity by simulating their steps. It takes into account the correct sequencing and other knowledge, such as the justification of the steps or the misconceptions. RIDES [Munro *et al.* 97] allows the generation of training systems focused on interactive graphic simulations. The author must define procedures comprised of a fixed sequence of actions, and the system in turn detects the student's actions, preventing malfunctions by means of a *model tracing* process. DEMONSTR8 [Blessing 97] describes the domain through production rules that represent the right knowledge, with the student's knowledge being represented as a Bayesian belief

network. It monitors each of the student's steps during task performance, and applies the model tracing technique, updating the network according to the diagnosis results.

Most of the authoring tools studied include a unique diagnostic mechanism that is closely related to the domain, and therefore they are of little value for different types of subject areas. Thus, in this work we aim to define generic diagnostic models, independent from the learning domains, in order to obtain a greater flexibility and portability to different domains.

2.3 A Starting Hypothesis

None of the described diagnostic techniques taken separately can fulfil our first goal of defining a generic diagnostic engine. But we claim that a combination of some of them could retrieve adequate diagnosis information to improve the student's learning process, and solve or reduce the problems showed by each of them. Therefore, we propose a hybrid and generalised diagnostic system that combines error libraries, the model-tracing technique, and a variant of the constraint-based modelling. In this approach, the teacher must define the learning domain by means of problem solving models, restrictions and bugs. These specifications will be used later when the student solves an exercise: the problem solving models will be the basis for the model tracing technique, while the bugs library and restrictions will show information about the most relevant and usual errors. Nevertheless, as the information defined by the expert may still be incomplete, we will also represent procedural knowledge including the prerequisites and postrequisites of each procedural action.

On the other hand, our second goal is to build an authoring environment to provide teachers with tools that enable them to create diagnostic learning systems based on sophisticated techniques on their own. Thus, they will define the domain knowledge requisites in a flexible and guided way, using the appropriate combination of techniques, so as not to force a unique knowledge representation schema.

3 *KADDET*: A Diagnosis Environment for Learning Tasks

KADDET is a developing environment oriented to the creation of *"learning by doing"* systems that allow for authoring and diagnosing processes. It has been conceived according to the following strategic purposes*: "Genericity"* to enable its instantiating to diverse domains; *"Suitable and sufficient diagnosis",* to be useful in the learning process; and *"Usability"* to encourage and favor its use.

The two functionalities described have led us to design and create two closely related systems, DETECTive and KADI, as well as a common shared theoretical basis—the Model for Learning Tasks. The latter allows for the description of the subject domain and the learning tasks from the perspective of the diagnosis of the student's knowledge. The next sections show the main components of *KADDET*. Nevertheless, for the sake of brevity, we will focus only on the procedural domains, which, being more complex, are more interesting than the conceptual domains.

3.1 Model for Learning Tasks. A Theoretical Basis

The Model for Learning Tasks (MLT)[1] defines an ontology [Welty 03] that identifies the components of the domain that need to be described in order to formalize the diagnosis process of students accomplishing procedural tasks. In addition, the model proposes the component interrelations, properties and scope. Keeping these needs in mind, the MLT defines elements for describing: (a) the procedural domain; (b) the learning tasks, or exercises; and (c) the results of the diagnosis process.

The domain is primarily defined by two items: the manipulatable *objects* and the *procedures* (or basic student actions). For the *objects*, the MLT establishes a minimal characterization, enabling the users to introduce new relationships and characteristics to define their peculiarities. Due to the application of procedures, an object takes different values or states during its lifetime. On the other hand, the specification of a procedure sets forth the states in which it can be executed (*prerequisites*), the variations produced by its performance on the objects of the scenario (*simulation actions*), and the new states of the objects involved (*postrequisites*). In order to define the procedures, the MLT provides the following elements: *parameter* of the procedure, *condition*, *graph*, *node*, *link* and simulation *action* (see Figure 1).

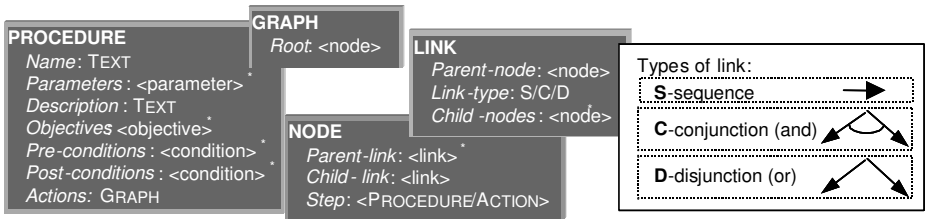

Figure 1: MLT elements for Domain Representation: Procedure

A learning task or Exercise [Almond *et al.* 02] allows variations and gaps in the learner's knowledge to be inferred. It defines its assessment criteria and is associated with some contents of the subject area by means of a set of learning objectives. In particular the *Practical Exercise* (Figure 2) describes the scenario to which learners will apply their knowledge, i.e. the set of domain objects that are suitable for manipulation at each moment of the problem solving process. In addition, the most frequent solving behaviors compose several *solution patterns*, and a series of recurrent or standard *Errors* identifies gaps in knowledge. According to these ideas, an MLT *Exercise* is defined by its presentation, the initial and final states of the scenario and a set of potential solutions. Additionally, the exercise includes pedagogical information about its difficulty, estimated time for completion, number of allowed attempts, and so forth. An exercise *Solution pattern* includes its evaluation and a resolution plan (be it right or wrong) that defines the sequence of steps (basic actions) to be taken. Our model distinguishes two types of errors: *Deviations* and *Predefined Errors*.

[1] MLT has been achieved on the basis of an empirical development and later refinements made to different domains. They are: *Derivation* in mathematics [Ferrero *et al.* 97], Photography- [Dorronsoro 93], Labour disability- *Help and Monitoring* in the work of mentally disabled people [Urretavizcaya *et al.* 99], the world of blocks [Ferrero *et al.* 99], and the industrial domain of the *Machine tool* [Lozano *et al.* 04].

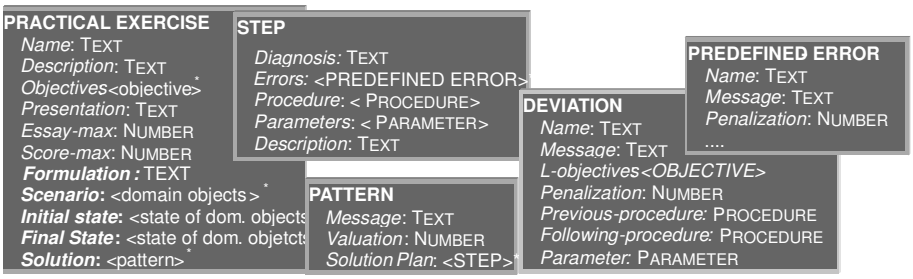

Figure 2: MLT elements for representing a learning task- practical exercise

Finally, the diagnosis result identifies those aspects relevant to the teacher, to the student or even to other support systems, with a focus on the *Learner's response,* which describes the solution given by the student as a linear sequence of resolution steps with information about the procedure performed, its diagnosis, the list of identified errors, and a numerical score.

3.2 DETECTive. A practical implementation

DETECTive implements the MLT ontology and bases its diagnosis on the MLT elements instantiated for a specific domain; it also carries out a Multiple Diagnosis Model based on different techniques. An example of a simple procedural domain inspired by *the blocks' world* will help us illustrate the main working ideas (Figure 3); its scenario is composed of a main box B0, with some cubic blocks (Bi) to be stored by a robot-hand by means of a set of procedures (Pick_Up, Leave_in_Box, Leave_on_Table, Pick_up&Leave_in_box).

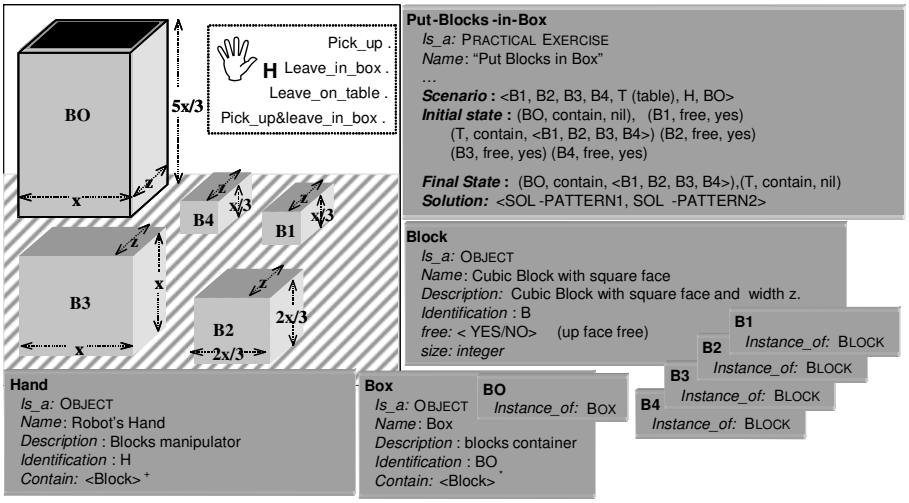

Figure 3: Scenario objects and MLT elements for a Robot Domain

The MLT ontology elements for domain definition (Figures 1 and 2) are organized at the *Abstract level*. The *Concrete level* instances the Abstract Level, characterizing the domain ontology with its manipulatable objects, procedures and exercises (Figure 3). The *Resolution Level* includes the exercise solution patterns that determine how the simulation actions associated with the domain procedures are to be performed. Figure 4 shows a part of each of the above described levels.

3.2.1 Diagnosis Model

Our proposal for the Multiple Diagnosis Model (MDM) applies a multiple process to the student's solution in order to detect its malfunctions and potential errors. The diagnosis of procedural exercises starts with a *Model-Tracing* treatment, which consists of searching for a solution plan that matches the learner's action. Since the domain description may be incomplete and the lack of solution plans relevant to the current exercise, the process incorporates other mechanisms that increase the number of recognizable potential actions: *Dynamic plan adaptation,* using information from the deviations and *Prerequisite verification* of the procedures identified in the domain. These mechanisms allow for the establishment of two complementary types of diagnoses: *Pattern-based Diagnosis,* through model-tracing and dynamic adaptation, and *Specific Diagnosis* centered on the prerequisite verification. As long as the learner's actions match the actions retrieved in a solution plan, DETECTive performs a Pattern-based Diagnosis; otherwise, it makes a Specific Diagnosis.

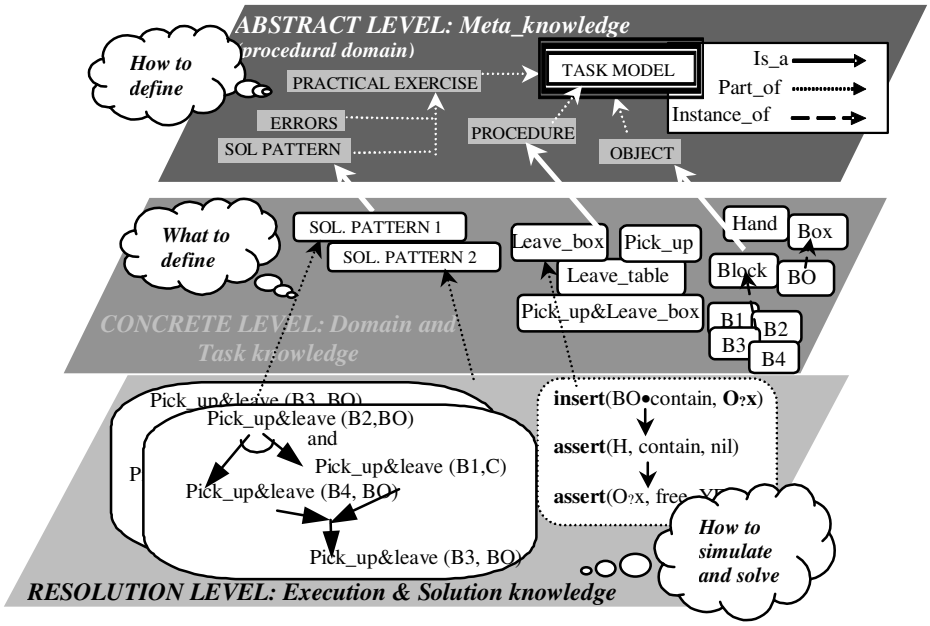

Figure 4: Elements of the Knowledge Levels

The *Pattern-based Diagnosis* relies on three main mechanisms: (i) monitoring of the Learner's Solution, as the procedure executed at each time of the problem solving process needs to be known; (ii) checking the solution plans included in the exercises, which tells which procedures are feasible at any time in order to determine the correction of the student's steps; and (iii) adapting plans through the domain deviations.

Initially, it considers all plans defined for the problem, i.e. Active Plans. The suitability of the learner's step is determined by comparing it with the information included in the follow-up item of the *Active Plans,* in such a way that the *Active Plans* that do not include the learner operation become *Rejected Plans*. When none of the plans reflects the student's action, the Dynamic Plan Adaptation proceeds. This process takes every previously *Rejected Plan*, and searches for a *Deviation* to explain the difference between the learner's step and the step described in the plan. If it succeeds, the *Rejected Plan* is restructured with the error-associated information and becomes a new *Active Plan*. Thus, a new pattern with a plan that fits the learner's response is available. Patterns whose rejected plans cannot be adapted become *Removed Patterns*. If upon completion of the monitoring of the current student's step, one or more plans remain active, it means that the step has been acknowledged and the Pattern-based Diagnosis goes on accordingly. If none of the *Rejected Plans* can be adapted, the Pattern-based Diagnosis finishes, the student's Solution tracing is suspended, and the Specific Diagnosis based on the domain knowledge is triggered.

The monitored acknowledgment of each student's step involves its simulation and diagnosis. On the one hand, the execution of the simulation actions of the procedure associated with the step changes the *Current State* of the scenario. On the other hand, the information about the step of the active plan (not adapted or adapted through an error), allows the retrieval of information for the diagnosis of the student's step.

The *Specific Diagnosis* verifies the learner's step applicability by comparing the current state of the scenario to the prerequisites of the procedure involved. So, if the prerequisites of the student's procedure-step are true, then it is simulated and diagnosed, and the process continues. Otherwise, the learner's response is wrong and the process stops. Thus, if the simulation of the learner responses reaches the exercise's final state, we will safely say that such a solution is complete and solves the problem.

3.2.2 Modular Architecture

DETECTive has been implemented via a modular architecture that allows it to be easily changed, widened and integrated with other systems. It is comprised of six modules (Figure 5): Diagnosis/Assessment Module, Functional Domain, Control Module, Simulator, Reporting Module and Student's Module.

The *Functional Domain* includes the domain-specific information, stored in two knowledge bases and one Error Catalogue, and implements the MLT Concrete and Resolution levels. The *Control Module* supervises and controls the diagnosis process: (a) proposing the exercise to the learner; (b) retrieving each solving step taken by the student and triggering the suitable diagnosis mechanisms; and (c) updating the diagnostic information on each step with its assessment (right, error, non-optimum ...) and the errors detected by the Diagnosis/Assessment Module. All these data complete the Diagnosis Result that makes up the Student's Module.

The *Diagnosis/Assessment Module* diagnoses each of the student's steps following the MDM shown in section 3.2.1. For this purpose, and once the student's step has been revised and acknowledged, it triggers the *Simulator* that carries out the procedure in terms of its simulation actions described in the Functional Domain. The *Student's Module* manages the information about the diagnosis of the exercises made by the learner. At present, it comprises a record of the diagnostic results. The *Reporting Module* is triggered from the *Control Module* upon completion of the diagnosis of one exercise, and generates a report of the student's resolution.

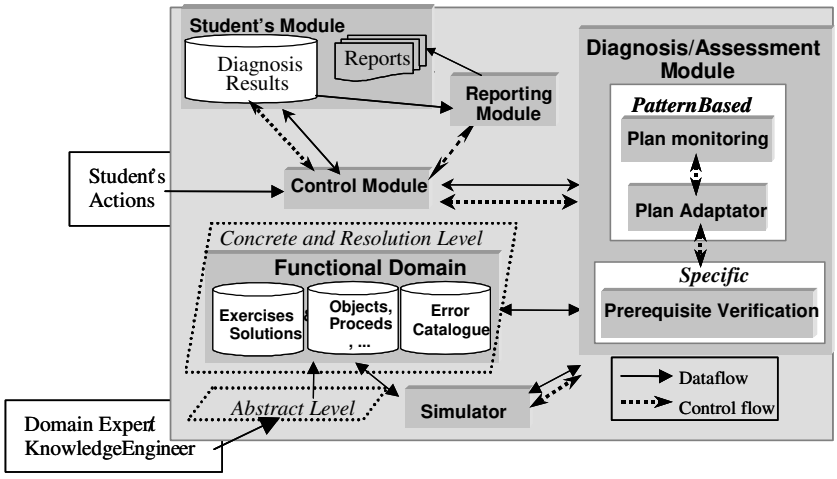

Figure 5: DETECTive's Architecture

3.3 Feeding the Model of Learning Tasks by means of KADI

KADI is the interface environment used to cover the process of knowledge acquisition of the *DETECTive diagnostic engine.* It includes two funcitonalities, teacher-*authoring* and student-diagnosis, which have been tackled generally by means of an interface system supported by a data management application. In this paper, we will address the authoring process, which allows the objects, procedures and exercises of the educational domain to be gathered, together with the correct and incorrect solving plans and their relevant errors.

The KADI architecture (Figure 6) is directly connected to the DETECTive engine. During the acquisition or authoring phase, KADI records the information provided by the teacher to feed the DETECTive's MLT (see section 3.1), and compiles it in CLIPS-syntax knowledge bases.

The acquisition process is carried out by means of three main modules. The *Exercises Acquisition Module (EAM)* allows both conceptual and procedural exercises to be defined. The *Object Acquisition Module (OAM)* defines the objects that the student has to manipulate during the procedural exercises. The *Procedures Acquisition Module (PAM)* defines the procedures that can be applied to the described domain objects. Upon the definition of the subject domain and the learning activities, two kinds of files are created with the information needed by the diagnostic engine. When DETECTive loads these files, the Concrete and Resolution Levels are created

completing the Functional Domain. KADI is supported by the DETECTive's diagnostic results during the definition of procedural exercises and the process is controlled by the *Guided Definition Management System (GDMS)*. In this situation the *Specific Diagnosis* is used to supervise, formalize and guarantee the completeness of the domain (at least for the described exercises). If one of the values indicated in the definition of a solution plan step (section 3.1) does not match the values expected in the simulation, the step is considered to be non appropriate. This is either because its characteristics have not been well defined, or because a previous step, necessary to reach a state allowing the current step to be applied, is missing. Finally, if the simulation of the created plan reaches the exercise final state, that solution is validated.

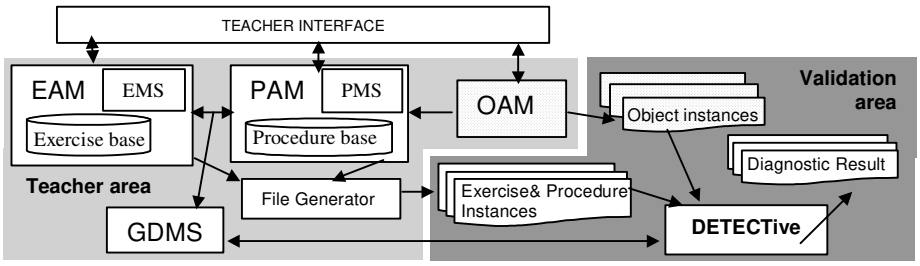

Figure 6: Architecture of KADI

The current version of KADI uses text window interfaces, whose design criteria—for instance, its "usefulness" and "simplicity" for users—were based on usability heuristics such as *Consistency* and *Error prevention* [Nielsen 93]. We have developed a methodology to provide teachers with a guide to introduce data in the right order. As a first step, the teacher identifies and creates the domain, and then the authoring proceeds by defining the exercises. A detailed description with illustrative examples is shown in [Martín *et al.* 04].

4 Conclusion. Lessons Learned and Future Work

In this paper, we have presented the *KADDET* system, an environment for cognitive diagnosis created to assess the conceptual and procedural learning activities of students, with a teacher-oriented authoring component. It was conceived on various bases: *Genericity* to allow its instantiation for different domains; *"Suitable and sufficient Diagnosis"* to make it a useful tool for supporting the student learning process in conceptual and procedural domains; and *"Usability"* to facilitate teachers' tasks of defining the subject matter and learning activities in order to promote and support its use.

KADDET is comprised of two integrated subsystems, DETECTive and KADI, sharing a common ontology Model of Learning Tasks. DETECTive is a generic, domain-independent diagnostic engine that implements a new diagnostic hybrid approach based on the discriminated integration of several paradigms. In this way, it

can be customized to a wide variety of conceptual and procedural domains. Since the correct behavior of the customized system depends on the correctness and completeness of the information provided in the acquisition phase, KADI facilitates the authoring process and controls the robustness of the input data.

DETECTive has been evaluated according to an adapted *pilot test* technique. The multiple paradigm diagnosis described in the system has proved to be sufficient to support learning in several domains, but some lessons have been learnt as well. Despite the flexibility of the tool, the creation of the domain, exercises and solutions is not an effortless task for teachers and, therefore, it demands some specific design skills. [Ferrero 04] presents a complete description of the validation tests performed in the context of three research projects and the results obtained.

Although we are well aware of the difficulty of achieving our initial goals, we have taken an important step forward toward those ends. Comparing our diagnostic proposal with the ones previously described, we share the opinion [Greer&Kohen 95] that a multiple paradigm allows teachers to choose the best technique, depending on their viewpoints and on the characteristics of the domain. Also, the combination of all paradigms results in a synergy, which exceeds the result of each individual paradigm.

Our current developments and future work focus on the improvement of DETECTive and KADI. Regarding DETECTive, we propose to validate and refine the multiple diagnosis process to better tackle the diagnosis complexity on procedural domains; in keeping with this line, new paradigms will be explored. On the other hand, we are redesigning and adapting its architecture to a multi-agent structure to facilitate both its integration in web-oriented applications and the addition of new diagnostic techniques. KADI is currently being updated, especially with regard to the description of the domain procedures and problem solving plans. Furthermore, a standard learning interface is under study to record and validate the student's answers, as well as to provide a means to display the diagnostic results and the general knowledge model. Finally, a new project involving virtual reality environments for procedural training is currently under development [Lozano *et al.* 04], aimed at treating screen representation of complex information and directly capturing the acquisition of knowledge, handling operations, and solving process in the virtual environment.

Acknowledgements

This research has been supported by the Spanish Ministry of Science and Technology MCYT (TIC-2002-03141) and the Univ. of the Basque Country (1/UPV 00141.226-T-13995/2001).

References

[Almond *et al*. 02] Almond, R.G., Steinberg, L.S., Mislevy, R.J.: "Enhancing the design and delivery of assessment systems: A four-process architecture". J.TLA (Journal of Technology, Learning, and Assessment) 1 (2002), 5, 2-63.

[Anderson et al. 90] Anderson, J.R., Boyle, C.F., Corbett, A.T. & Lewis, M.W.: "Cognitive Modelling and Intelligent Tutoring". J.AI (Journal of Artificial Intelligence), 42 (1990), 7-49.

[Anzai & Simon 79] Anzai, Y., Simon, H.A., "The theory of learning by doing". Psychological Review, (1979), 86, 124-140

[Burton 82] Burton, R.B. "Diagnosing bugs in a simple procedural skill". Intelligent Tutoring Systems, Sleeman, D. & Brown, J.S. (Eds.), New York: Academic Press, (1982).157-183

[Baffes *et al*. 96] Baffes, P., Mooney, R.:" Refinement-Based Student Modeling and Automated Bug Library Construction".J.AIED (J. of Artificial Intelligence in Education), (1996),7, 75-117.

[Blessing 97] Blessing, S.B.: "A Programming by Demonstration Authoring Tool for Model-Tracing Tutors". J.AIEd (J. Artificial Intelligence in Education), 8 (1997), 233-261.

[Dorronsoro 93] Dorronsoro, I.: "Sistema Tutor Inteligente para la enseñanza de la fotografía". MSc dissertation. University of the Basque Country UPV/EHU, Donostia (1993).

[Ferrero 04] Ferrero, B.: DETECTive: un entorno genérico e integrable para diagnóstico de actividades de aprendizaje. Fundamento, diseño y evaluación [Tesis Doctoral]: Univ. of the Basque Country UPV/EHU (2004).

[Ferrero *et al*. 97] Ferrero, B., Fernández-Castro, I. & Urretavizcaya, M.: "DETECTive: A Generic Diagnostic Tool to Support Learning. Some experiences in the symbolic differentiation domain". Proc.CAEE'97, Krakow, Poland (1997), 54-61.

[Ferrero 99] Ferrero, B., Fernández-Castro, I. & Urretavizcaya, M.: "Using DETECTive, a generic Diagnostic System for procedural domains". Proc. AI-ED'99.(1999), 667-669.

[Goldman *et al.* 99] Goldman, P., Geib, C.W. & Miller, C.A. "A New Model of Plan Recognition". *Conf. on Uncertainty in Artificial Intelligence* (1999). http://www2.sis.pitt.edu/~dsl/UAI/UAI99/main.pdf

[Greer & Koehn 95] Greer, J.E. & Koehn, G.M. "The Peculiarities of Plan Recognition for Intelligent Tutoring Systems". WS *The Next Generation of Plan Recognition System.* (IJCAI'95), (1995) http://www.dfki.de/~bauer/workshop.html#schedule

[Hsieh *et al*. 99] Hsieh P., Half, H. & Redfield, C.: "Four easy pieces: Developing systems for knowledge-based generative instruction". J.AIEd (J. Artificial Intelligence in Education), 10 (1999), 1-45.

[Joolinge *et al*. 96] Joolingen, W.R. & de Jong, T.: "Supporting the authoring process for simulation-based discovery learning". Proc. ECAI'96, Budapest, Hungary (1996), 66-73.

[Katz *et al*.94] Katz, S., Lesgold, A., Eggan, G. & Gordin, M.: "Modeling the Student in Sherlock II". Proc. NATO Advanced Research, Springer-Verlag, Berlin (1994), 99-125.

[Kono *et al*. 94] Kono, Y., Ikeda, M. & Mizoguchi, R.: "THEMIS: A nonmonotonic inductive student modelling system".J.AIEd (J. Artificial Intelligence in Education),5, 3 (1994),371-413.

[Lozano *et al*. 04] Lozano, A., Urretavizcaya, M., Ferrero, B., Fernández-Castro, I., Ustarroz, A., Matey, L.: "Integration of a Generic Diagnostic Tool in Virtual Environments for Procedural Training". LNAI 3040 (2004), 666-675.

[Martin *et al*. 04] Martín, M., Fernández-Castro, I., Urretavizcaya, M., Ferrero, B.: "KADI: A Knowledge Acquisition Tool for Learning Activities". Proc. Ed-MEDIA'04. Lugano, Switzerland (2004), 889-894.

[Millán *et al.* 00] Millán, E., Pérez-de-la-Cruz, J.L., Suárez, E.: "Adaptive Bayesian Networks for Multilevel Student Modelling". Proc. ITS'00, Montréal, Canada (2000), 534-543.

[Mitrovic *et al.* 99] Mitrovic, A., Ohlsson, S.: "Evaluation of a Constraint-Based Tutor for a Database Language". J.AIEd (J. Artificial Intelligence in Education), 10, (1999), 238-256.

[Munro *et al*. 97] Munro, A., Johnson, M.C., Pizzini, Q.A., Surmon, D.S., Towne, D.M., Wogulis, J.L.: "Authoring Simulation-Centered Tutors with RIDES". Int. J. of Artificial Intelligence in Education, Vol 8 (1997), 284-316.

[Murray 97] Murray, T., "Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems". J.AIEd (J. Artificial Intelligence in Education), (1997), 8, 3-4, 222-232

[Murray 03] Murray, T., "An overview of Intelligent Tutoring Systems Authoring Tools". In Murray, T., Blessing, S., Ainsworth, S.(Eds.): Authoring Tools for Advanced Technology Learning Environments, 491-544. Kluwer Academic Publishers (2003).

[Murray *et al.* 03] Murray, T., Blessing, S., Ainsworth, S.: "Authoring Tools for Advanced Technnology Learning Environments"; Kluwer Academic Publishers (2003).

[Ohlsson 94] Ohlsson, S. "Constraint-Based Student Modelling". Greer, J.E. & McCalla G.I. (Eds.) *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, Berlin: Springer-Verlag, (1994), 167-189.

[Nielsen 93] Nielsen, J.. "Usability Engineering"; Academic Press, Inc. (1993)

[Suraweera & Mitrovic 04] Suraweera, P., Mitrovic, A., "An Intelligent Tutoring System for Entity Relationship Modelling". J.AIEd (J. Artificial Intelligence in Education), (2004), 14, 3-4, 375-417.

[Urretavizcaya *et al.* 99] Urretavizcaya, M., Ferrero, B. & Fernández-Castro, I.: "Visión general de TUTOR. Las personas con deficiencias cognitivas se integran en el entorno laboral". J. Boletín Digital Factors Humans, 21. http://boletin-fh.tid.es/bole21/art002.htm.

[Welty 03] Welty, C.: "Ontology Research". J. AI Magazine, 24,3 (2003), 11-12.