ELSEVIER Microprocessing and Microprogramming 40 (1994) 855–860

Microprocessing and Microprogramming

MeT: The Expert Methodology Tutor of GENITOR

I. D. Zaharakis [a,c] A. D. Kameas [b,c] P. E. Pintelas [a,c]

[a] Division of Computational Mathematics & Informatics, Department of Mathematics, University of Patras, Greece

[b] Department of Computer Engineering & Informatics, University of Patras, Greece

[c] Educational Software Development Laboratory, Department of Mathematics, University of Patras, Greece

In this paper, the domain expert system, that a generator of intelligent tutoring systems (ITS) includes with the applications it produces, will be presented. This system, MeT (Methodology Tutor), is responsible for the description and teaching of the procedural domain knowledge (called methodology) of the ITS. The knowledge handled by the system is represented with frames. MeT operates in two modes: during authoring mode, several editors are provided to support the authors in the description of the methodology that will be taught. In tutoring mode, three agents are provided: a guide for the simulation of the methodology evolution, a learn-by-discovery agent that comments on the students' actions and a judge that validates the students' selections. Finally, a part of the procedural knowledge of an example application is presented.

1. INTRODUCTION

GENITOR [6] is a generator of intelligent tutoring systems (ITSs) that use coaching and simulation techniques for knowledge transfer. Applications developed with the system employ two expert systems for the transfer of declarative and procedural domain knowledge [1].The form in which this knowledge is stored determines the ways it can be used. No general form suitable for representation of all types of knowledge exists [10]. Of the several knowledge representation schemes that have been proposed in the past, production rules and frame networks are the most popular ones. Production rules have been widely adopted for domain knowledge representation [1,3], both because they are easy for humans to understand, since they come close to the human way of reasoning, and can provide rudimentary explanations of the system reasoning process. Frames are an equivalent (in representation terms) way of grouping information. Each frame is a record of "slots" and "fillers" [7] and can be thought of as a complex node in a network. Frame systems use abstract representations of knowledge in order to reason about classes of objects.

In this paper, the design of an expert system for teaching procedural knowledge is presented. The procedural knowledge in GENITOR terms is called "methodology", and the presented expert system is the Methodology Tutor (MeT). MeT employs simulation to allow for learning in a problem-solving context, enabling the application of this knowledge in real-life. Simulation-based domains are usually represented with object networks in association with rules and

0165-6074/94/$07.00 © 1994 – Elsevier Science B.V. All rights reserved.
SSDI 0165-6074(94)00068-9

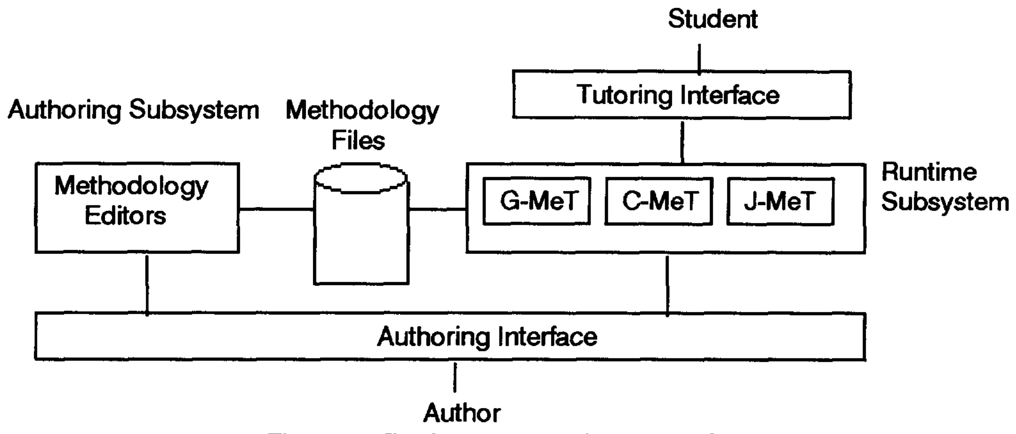

Figure 1: The functional architecture of MeT

constraints [2,4].Their problem is mainly the difficulty of integration with the other components of the tutoring system, although this did not come up during GENITOR design.

In the next section, the architecture of MeT, together with the adopted knowledge representation schemes will be presented. Then, the system-user interaction in both operation modes (authoring and tutoring) is described. An example knowledge base is subsequently described, and the paper concludes with the future research directions of the authors.

2. THE METHODOLOGY TUTOR (MET)

A methodology is any procedure that consists of distinct, partially ordered tasks, actions to carry out each task and results of each action [8]. Such a methodology has a well-defined structure that consists of several layers of groups of activities (phases and subphases), activities that make up a group and control activities that mark the termination of a group. With each activity, artifacts are produced; these have the meaning of inputs (prerequisites) or outputs (objectives), respectively, of the activity. In order for an artifact to be produced, other artifacts must be already existing (some of them may even be consumed). In this way, a partial ordering in the form of a dependency and precedence graph of the activities is derived by MeT, which also ensures that at least one deadlock-free traversal of the graph exists.

2.1 Architectural design

MeT consists of two subsystems (Figure 1): the authoring subsystem, which provides the tools that the authors can use in order to describe the methodology to be taught, and the runtime subsystem, which is used for the presentation and tutoring of the methodology.

2.2 Knowledge representation

The methodology elements manipulated by MeT are represented with frame objects. These are instantiations of classes that are structured according to the inheritance schema of Figure 2.

All kinds of frames have a class and an is-a field, which store the name of the class and its parent in the inheritance relationship. The values of these two fields are fixed and can not be changed. Instead, slots are used to store changing information. All classes have three attribute slots in common: *identifier,* which indicates the author-defined name that the class assumes in the methodology which the model instantiates, *label,* which identifies the author-defined instantiation of the class object,

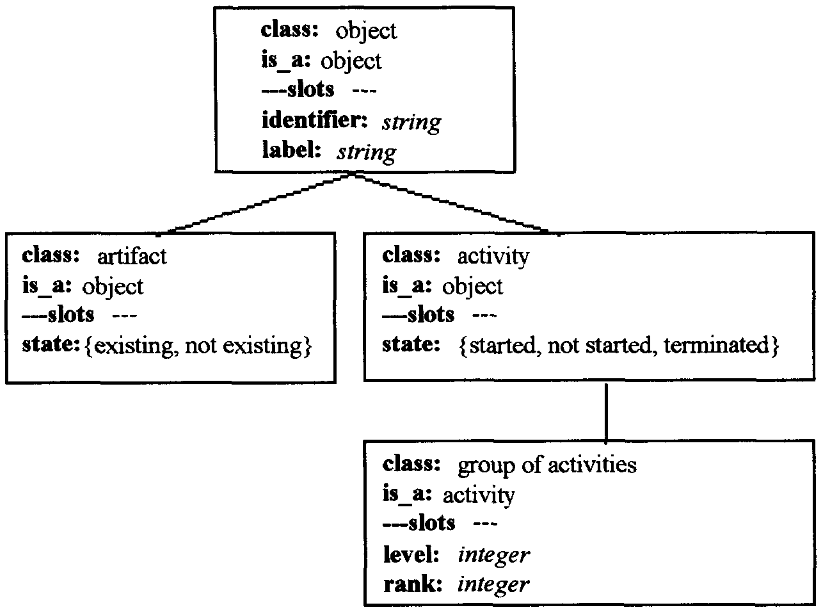

Figure 2: The inheritance schema and the structure of the frames used by MeT

and *state,* which indicates the state of the class. The value of the latter slot is default in the beginning but is changed by the system along the simulation. Permitted slot values for the classes activity and group of activities are "not started", "started" and "terminated"; for the class artifact are "not existing" and "existing".

The class group of activities has the additional attributes *level* and *rank*. The values of the *level* and *rank* slots are numbers ranging from 0 to n, $n \in N^*$. These values are specified by the authors during the authoring process and cannot be changed during the simulation. The *rank* slot indicates explicit ordering between several groups of activities. Use of explicit ordering is optional, since authors may implicitly specify ordering of groups of activities with the appropriate use of artifacts. The *level* slot offers the authors the ability to create groups and subgroups of activities.

3. SYSTEM-USER INTERACTION

Two classes of users, each with different characteristics and requirements, interact with MeT: authors and students (trainees). Different operation modes and different interfaces are provided for each class. Both interfaces adopt common interaction metaphor features which have been described with the IMFG model [5].

3.1 Authoring mode

MeT supports either top-down (phases, subphases, activities, artifacts) or bottom-up (that is, starting from the artifacts) methodology specification. A mixed process may be adopted as well. Several functions that enable the authors define the structure and explicit ordering of activities and groups of activities, the artifacts each activity produces and the artifacts each artifact needs or consumes in order to be produced. In addition, several authoring functions are provided to support authors during methodology validation. These provide

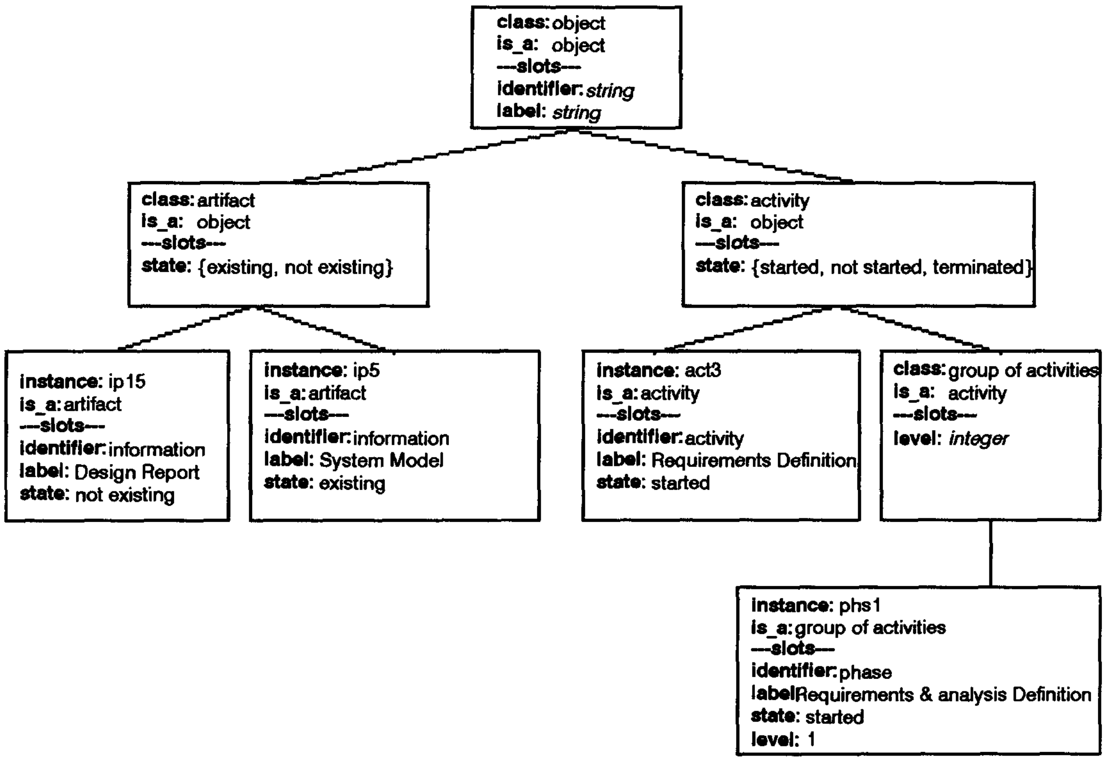

Figure 3: The instances of the frames used in the example

answers on whether a group of activities can be started, which activities it contains and whether these can be started or executed, and in general, whether a methodology element can reach a certain state or be produced at all. In any case, MeT supports a prototyping facility that enables authors to test the appearance and efficiency of the methodology under development.

In order to validate the design of the methodology, the system uses internally a Petri Net-based model. Specifically, the methodology structure is transformed into an Individual Token Net, with a general high-level Petri Net model. Activities and control activities correspond to transitions, while artifacts are modeled with places. The advantages of this approach are that well-documented analysis methods (like coverability tree, incidence matrix etc) can be used to ensure that the described methodology is deadlock-free and live [9].

3.2 Tutoring mode

During tutoring mode, the runtime subsystem of MeT presents the student with a simulation of the methodology to be taught. The dialog is based on problem solving: the problem posed each time is to find the correct activity that must be taken next in order to advance inside the methodology structure. The system or the student must choose among the set of all activities of the methodology. During this mode, MeT may employ one of the following three agents: guide, coach, judge.

The task of MeT-Guide (G-MeT) is to simply present the procedural knowledge to the students. To this end, it "guides" them through a simulation of the methodology evolution by selecting itself each time the correct activity. The progress within the methodology is visualized using animated charts and diagrams. The system presents the valid alternatives at each step, and also produces a description of the new state. The students have several operations at their disposal, like pause, continue and stop to control simulation flow, move to, in order to select the (future or past) starting point of the simulation, where am I, to get a description of the current state, why, to get justification of the system actions, inventory, to see the names and quantities of the artifacts been produced so far.

MeT-Coach (C-MeT) supports a learn-by-discovery process of the methodology. It is the students who must now select the next activity from the set of all methodology activities. The role of C-MeT is to guide the students by commenting on their selections, responding to their commands and assuming control when they appear to be lost. This mixed-initiative control scheme is based on an encoding of students mistakes, which, combined with the student model, permits a precise diagnosis of students misconceptions and of the adaptation of the remedy strategy to their requirements and capabilities. Operations "pause", "continue", "stop", "inventory", "where am I" and "why" are also supported in this mode, while the verbose level is decided by the system itself after consulting the student model. In place of command "move to", C-MeT offers commands "what if", which appears when more than one alternatives exist for the next step, and which permits the students to explore all possible alternatives (by conducting a simulation of the possible futures of the methodology simulation), and "show me", which is a call of the students to C-MeT for the presentation and justification of the correct solution.

Finally, MeT-Judge (J-MeT) leaves control of the simulation entirely at students hands, by adopting a role of "judge" of their selections. J-MeT only informs students on the validity of their selections, using messages of minimal semantic content. The only available command is "inventory", but, depending on authors choice, commands "pause" and "background" may also be available.

4. AN EXAMPLE

As an example, in Figure 3, a tree-like frame structure is shown for a part of the domain knowledge of an application developed with GENITOR that teaches the Waterfall Model of Software Engineering. This methodology consists of five steps called "phases" (a phase is modeled with a frame that *is-a* group of activities having slots *identifier*=phase and *level*=1): Requirements Analysis & Definition, System & Software Design, Implementation & Unit Testing, Integration & System Testing, and Operation & Maintenance. The current phase is Requirements Analysis & Definition (as its *label* slot states); its *state* slot has the value "started".

The activity being carried out, which also belongs to this phase, is Requirements Definition. The frame that represents this activity has *state*="started". As far as artifacts are concerned, two of them are represented: one, System Model, has been produced during a previous activity (thus, its *state* is "existing"), and the other, Design Report will be produced by a future activity (thus, its *state* is "not existing").

All the slots in the frame instances have a certain value that is unique for each instance. While the author decsribes the methodology that will be taught, the system automatically constructs the inheritance scheme and fills the appropriate slots. The values of certain slots change with execution of the methodology (i.e. slot *state*).

5. CONCLUSIONS

A part of the domain expert system of GENITOR that is responsible for the tutoring of procedural knowledge has been presented. MeT cooperates with the declarative knowledge tutor of GENITOR (DeT), which is responsible for the presentation of units of learning material that support the application of procedural knowledge. To this end, a special operation (called background) is included in MeT tutoring interface. By using this operation, students can combine both MeT and DeT during methodology simulation and get a list of topics of declarative knowledge proposed by the system which relate to the current simulation state.

Extensions that system authors are considering include the provision of a flexible structure that will enable authors describe procedural knowledge of differing internal structure and the use of an artificial neural net for student modeling. In addition, the provision of more agents during tutoring mode is considered.

REFERENCES

1. J.R. Anderson, C.F. Boyle, A.T. Corbett and M.W. Lewis, Cognitive Modelling and Intelligent Tutoring. In Artificial Intelligence and Learning Environments (W.J. Clancey and E. Soloway, eds), MIT Press (1990), pp 7-49.
2. J.S. Brown, R.R. Burton and W.J. Clancey, Pedagogical, natural language and knowledge engineering techniques in Sophie I, II and III. In Intelligent Tutoring Systems (D. Sleeman and J.S. Brown, eds), Academic Press (1982), pp 227-282.
3. W.J. Clancey, Methodology for building an Intelligent Tutoring System. In Artificial Intelligence and Instruction (G.P. Kearsley, ed). Addison-Wesley (1987), pp 193-227.
4. H. Hinje and J. van Berkum, A functional architecture for intelligent simulation learning environment. In Learning Technology in the European Communities (S.A. Cerri and J. Whiting, eds), Kluwer Academic Publishers (1992), pp 585-593.
5. A. Kameas, S. Papadimitriou, P. Pintelas and G. Pavlides, IMFG: an interactive applications specification model with phenomenological properties. Proceedings of the 19th EUROMICRO Conference, Barcelona, Spain, September 6-9, 1993.
6. A. Kameas and P. Pintelas, GENITOR: a GENerator of Intelligent TutORing applications. Technical Report TR 94-01, Division of Computational Mathematics & Informatics, Dept. of Mathematics, Univ. of Patras, Greece (1994).
7. M. Minsky, A Framework for Representing Knowledge. In Psycology of Computer Vision (P. H. Wiston, ed), MIT Press, Cambridge, Mass (1975).
8. P. Pintelas, A. Kameas and M. Crampes, Computer–based Tools for Methodology teaching. Proceedings of the 34th International ADCIS Conference: Empowering people through Technology, Norfolk, USA, November 8-11, 1992, pp 341-355.
9. W. Reisig, A primer in Perri Net design. Springer-Verlag, (1992).
10. J.W. Rickel, Intelligent Computer-Aided Instruction: a survey organized around system components. IEEE Trans. on Systems, Man and Cybernetics, 19(1), (1989), pp 40-57.