

## THE ARCHITECTURE OF AN AUTHORING SYSTEM IN SMALLTALK/Y

E. Fischetti\*, A. Gisolfi\*.

## ABSTRACT

This paper discusses the problem of furnishing knowledge about different domains to a "shell" and describes an authoring facility implemented in Smalltalk/Y on an IBM-PC. Considerations involving the planning and construction of the shell are presented as regards both its internal and external architecture. The peculiar capabilities of Smalltalk/Y supply the basis for the environment of the shell.

## INTRODUCTION

The popularity of "shells" for Intelligent Tutoring Systems (ITS) is growing but, among teachers, there has been widespread dissatisfaction with most shells that they have used. We consider shells as potentially useful classroom tools, yet many problems have to be solved for they could develop into regular tools. In fact, we note that most researchers distinguish four modules in ITS (ref 8): A domain expert, that possesses the knowledge to be taught; a student expert, that suits the teaching strategy to the each student; a teaching expert, that explicates the tutorial strategy; an interface, that manages the dialogue between the user and the system.

Although there is a wide agreement about the general structure of ITS, the task of developing useful and effective systems is a challenging one: the problems in determining how to represent the knowledge domains, how the students learn in a given domain, are all topics which are still being investigated. Very few systems are, to date, truly operational and most of them are still prototypes that require improvements and extensive field-testing. Moreover, few tutoring systems attempt to operate across a set of domains ("shells"); in fact, these systems present, for instance, the problem of building the knowledge base and this usually is a heavy burden for individuals who are not programmers, but are only expert in the domain to be taught. Yet, we note a tendency towards standard architectures for ITS, in particular "shellification" as opposed to specialized individual ITS (ref 6, 5, 7, 1, 3). On the other hand, nowadays Object Oriented Programming (OOP) represents one of the most favourite areas among the emerging software technologies (see ref 4, 2 for OOP educational implications). The construction of a shell, in our opinion, can greatly benefit from a language like Smalltalk/Y: in fact, each concept or misconception is implemented as an object. Objects are hierarchically organized so that we have the inheritance of attributes needed for the diagnosis of conceptions and misconceptions. Moreover, visual access to each object is provided by the programming environment of Smalltalk/Y. Finally, Smalltalk/Y operates on IBM-PC and thus only quite inexpensive machines are required. The above sketched reasons suggested us developing an authoring facility which fully exploits the peculiar capabilities of Smalltalk/Y. Our prototype is capable of relevant improvements and, at the time of writing, it should not be regarded as an attempt to provide "intelligent" tutoring since our main objective was to build quickly an effective tool to explore the potential of OOP educational environment.

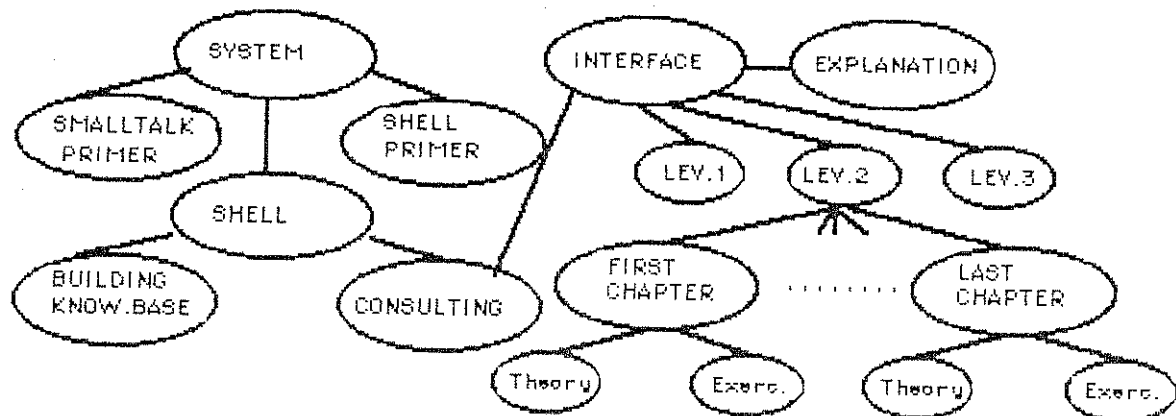
## THE SHELL

The two basic uses of shells are building, where users construct their own knowledge bases, and consultation, where, after building, the system develops the tutoring strategy. Our shell was thus designed as a tool with two basic environments (one for building and one for consulting) and several accessory tools for such tasks as explanation, browsing, practice, etc. We will now go on to describe the shell as regards both the internal and the external structure, for the expert and the end-user.

\*Dipartimento di Informatica, Università di Salerno, 84100 Salerno, Italy.

## INTERNAL ARCHITECTURE

A tree can be used to represent the architecture of the system (see figure); the nodes denote the constituting modules. First we encounter the root which represents the whole system: it has three sons: Smalltalk primer: personalizing the shell requires an elementary knowledge of Smalltalk/V, if the user does not possess it this module provides the basic notions necessary to accomplish this goal.



Both theory and exercises are present to attain quickly a sufficient mastery of the language.

Shell primer: a simple guide, in two parts, for using the shell; the first part addresses the end-user, the second addresses the expert in the domain to be taught to allow him building the knowledge base. The shell itself.

In turn, the shell was organized as follows. The nodes represent: a) Building the knowledge base: this module takes care of introducing the knowledge into the domain representation. The expert is supposed to be not a programmer and he is led to accomplish the building. b) Consultation: when the knowledge is introduced the tutor can be started. The structure is as follows: Explanations: current informations about the options offered by the menu. Interface: this module is the core of the system and manages the dialogue between the user and the system. Besides, it controls the modules representing the three instructional levels (beginner, intermediate, advanced) available in the tutor. The selection of the levels can be performed by the tutor.

Each level has as many sons as are the chapters it is subdivided into; each chapter has two sons:

- Theory: the underlying theory regarding the knowledge domain is shown
- Exercises: the mastery of the student is verified by means of suitable exercises.

## IMPLEMENTING THE SHELL

The hierarchical organization of the modules was implemented thanks to the capabilities of Smalltalk to define classes of objects. A class is associated with each module, each possesses the methods apt to implementing module functions. Besides, since Smalltalk has the peculiarity of class inheritance, we were able to organize the system according to the architectural hierarchy, so that modules and sub-modules of the system get associated with classes and sub-classes of Smalltalk/V. The methodological choice of fully exploiting the peculiar characteristics of Smalltalk/V language influenced also the guidelines for implementing the interface: we defined some suitable windows each associated with a corresponding menu so that a friendly interface is achieved. The user can select the appropriate entry in the menu or, alternatively, when the system behavior depends on user's answers to the question put by the system, the prompter can be used.

This typical Smalltalk/V structure is a little window containing the question to be asked the student and the possible answers. If the student answers correctly the related choice will be implemented.

The prompter is a powerful visual tool since it offers the informations in an attractive way.

The module interface branches off to the three sub-modules managing the teaching levels. These sub-modules are mutually independent, yet they are implemented via sub-classes of the same class Interface. The independence guarantees that no confusion can arise among the levels contents and it is achieved by assigning a different class to each teaching level; moreover, each level is labelled with an identifier that marks all the files containing informations about the level itself.

Finally, we note that the exercises associated with each chapter are dynamically organized by means

of a circular queue (implemented by "Ordered collection") that allows the expert inserting exercises at will and, furthermore, permits the student to leave temporarily an exercise and, later on, to resume it.

#### EXTERNAL ARCHITECTURE

The first approach to the system allows the user choosing one of the courses already present in the system: "Smalltalk primer" or "Shell primer". Both courses share the same structure and provide basic notions about Smalltalk/V (for personalizations) and the shell (for building), respectively. The tutorial strategy first furnishes the theoretical background, subdivided into chapters and then the related exercises are available to the user for the necessary practice. The same holds for the user of the tutor. The building of the knowledge base is performed by the expert by means of suitable windows and menus so that the aim is, step by step, achieved. In no way the operation can be unsuccessful: in fact, the expert is carefully guided by the system during the insertion of the knowledge. Then the tutor is ready. A window subdivided into two parts is first shown:

Upper part: contains the chapter currently under examination; it is list-pane like, so we select the item. Lower part: it is text-pane like and generally contains a text (e.g. a method, exercises, etc.). If the user selects a chapter among those shown in the upper part a menu will be associated with the lower one permitting the user to enter the desired section (theory or practice). This technique, incentrated on the couple of active windows, allows the user easily re-examining previous lessons. In particular, if the user wishes to practice, then the exercises associated with the selected chapter will be shown in the lower part, together with the answers, and the prompter is made available.

Should the answer be wrong, then the system, via menu, offers three different help levels:

Theoretical: summarizes the theory underlying the exercise. Hint: depending of the answer provides a suitable hint. Direct: answers correctly the question.

We note that there is no upper limit to the number of exercises and the same holds true for the number of answers. Moreover, the three instructional levels permit to achieve gradually the desired mastery.

#### CONCLUDING REMARKS

The OOP paradigm well reflects, in our opinion, the conceptual framework and permits to use a hierarchy of objects associated with the different types of concepts. The prototype developed by us is undergoing field-testing in various areas (e.g. elementary algebra, programming languages, geometry) and the preliminary results are, according to our classroom experiences, rather encouraging; yet we are aware that many constraints on the design of the shell (e.g. size, agility) lead us to privilege some features (e.g. effectiveness, friendliness) and prevented us from fully scrutinizing some important problems (e.g., student model, natural language interface). The research is under way and next releases will incorporate other advanced features; it is our opinion, however, that OOP will play, in the very near future, a more and more important educational role thanks to its peculiar characteristics.

#### REFERENCES

1. Breuker, J.A., Winkels, R.G.F. & Sandberg, J.A.C.: A shell for intelligent help systems, Proc. 10th IJCAI, p.167, 1987
2. Esposito, F., Lanza, A. & Roselli, T.: An authoring system for Intelligent CAI, in De Blasi, M. et al (Eds.): Education and application of computer technology, Bari, p.515, 1988
3. Fischetti, E. & Gisolfi, A.: LOGPRIMER: Building a tutoring system for Prolog learning, Tech. Rep. DIA15, Dip.Informaticá, Univ. Salerno, 1989
4. O'Shea, T.: Magnets, martians and microworld: learning with and learning by OOPS, Proc. 4th Int. Conf. AI & ED, Amsterdam, p.193, 1989
5. Scherz, Z., Weinberg, B., Pontch, M. & Goldberg, D.: EESS: Education Expert System Shells, Pegboard, 3, 1, p.100, 1988
6. Spensley, F. & Elsom-Cook, M.: Generating domain representations for ITS, Proc. 4th Int. Conf. AI & ED, Amsterdam, p.276, 1989
7. Valley, K.: Realising the potential of expert system shells in education, Proc. 4th Int. Conf. AI & ED, Amsterdam, p.307, 1989
8. Wenger, E.: Artificial intelligence and tutoring systems, Los Altos, Morgan-Kaufmann, 1987