USG
University of Strathclyde, Glasgow

Diogene

Survey on Methods and Standards for Student Modelling

Version 1.3

Revision History

Date	Version	Description	Author
27/05/2002	1.0	Initial version.	CRMPA
28/06/2002	1.1	Intermediate version. 4.4 paragraph added.	Ruth Wilson and Robert Villa, USG
30/07/02	1.2	Intermediate version. Paragraphs 3.2, 3.4, 3.6, 3.7, 3.8 added.	Robert Villa, USG
9/08/02	1.3	Final version. Section 3.4 rewritten. Section 2.4.1 altered	Robert Villa, USG

Table of Contents

1 INTRODUCTION 4

1.1 PURPOSE 4
1.2 SCOPE 4
1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS 4
1.4 REFERENCES 5
1.5 OVERVIEW 6

2 STANDARD MODELLING TECHNIQUES 7

2.1 INTRODUCTION 7
2.2 THE STUDENT MODEL 7
2.3 USER BEHAVIOUR AND THE SYSTEM BACKGROUND KNOWLEDGE 8
2.4 BUILDING STUDENT MODELS 9
2.5 BACKGROUND KNOWLEDGE FOR STUDENT MODELLING 10
2.6 STUDENT BEHAVIOUR 11
2.7 DOMAIN COMPLEXITY 11
2.8 TWO SCHOOLS OF THOUGHT 11

3 SOME SYSTEMS OF INTEREST 13

3.1 INTRODUCTION 13
3.2 SCHOLAR 13
3.3 DEBUGGY 15
3.4 LISP TUTOR 16
3.5 ASSERT 17
3.6 ANIMALWATCH 19
3.7 GRANT 20
3.8 I^3R 22
3.9 A BRIEF DISCUSSION OF THE PRESENTED SYSTEMS 23

4 LEARNING STANDARDS 25

4.1 AN INTRODUCTION TO LEARNER MODEL STANDARDS 25
4.2 IEEE P1484.2 (PAPI LEARNER) 25
4.3 IMS LEARNER INFORMATION PACKAGE (LIP) 30
4.4 SABA PROFILE FORMAT 37
4.5 SOME E-LEARNING SYSTEMS 41
4.6 OTHER INITIATIVES 42

Survey on Methods and Standards for Student Modelling

1 Introduction

1.1 Purpose

The DIOGENE Project aims to design, implement and evaluate a training Web brokering environment geared toward ICT professionals and able to cover the whole lifecycle of ICT vocational training inside and outside enterprises' boundaries. It will design and implement a self-learning Web environment able to support the individual from the definition of objectives to the assessment of the results through the construction of custom self-adaptive courses. The system will be accessible through the Web and will take the form of a training portal.

DIOGENE will provide learners with a system able to suggest optimal learning objectives, to determine their profile and the actual knowledge they have acquired, to dynamically assemble courses based on individual training needs and learning styles, and to join freelance teachers able to provide guidance and motivation.

We intend to provide DIOGENE with a learner model composed essentially of two different structures: a *Cognitive State* and a *Learning Profile*. In every instance the system will be able to update both structures for each learner on the basis of his (or her) development activities.

With the *Cognitive State* we intend to enclose all information about the level of knowledge reached by a particular learner about concepts of the domain covered. We plan to represent logically this information exploiting a set of fuzzy numbers (one for each concept). The decision to use fuzzy numbers in cognitive states arises from the necessity to manage the learner evaluation. In this way, we can admit different kinds of evaluations with different degrees of reliability.

Within the *Learning Profile* we intend to enclose all information about the learner's perceived capabilities (i.e. to which kind of resources a specified learner is shown to be more receptive) and preferred style of learning. This information includes (but will not be limited to) preferences about the following set of variables: kind of media, pedagogical approach, interactivity level, semantic density, difficulty, and so on.

All the information calculated by DIOGENE about learner models will be exploited during the intelligent course tailoring procedure. Moreover, learner models will be exportable in a standard format that will be individuated in order to allow the possibility of creating a learner CV to be published, with respect to privacy requirements.

The purpose of this document is to review current methodologies and standards in the field of student modelling. The author will try, with this document, to guide the work around the definition of a methodology for student modelling to be applied in the development of the DIOGENE Student Model.

This document will also suggest proper standards to be applied. For this reason, the principal standards related to student representation (PAPI, LIP, etc.) will be described and compared in order to try to select the best one to apply.

1.2 Scope

This document is related to another survey to be produced by USG in the DIOGENE context, the *Survey on Methods, Standards and Tools for LO Knowledge Representation.*

Together, the two surveys will inform the project's *Student and Knowledge Model*.

1.3 Definitions, Acronyms, and Abbreviations

See Glossary [1].

1.4 References

[1] CRMPA DIOGENE Glossary, 2002.

[2] The IEEE LTSC PAPI specification. Available at http://ieee.ltsc.org/wg2

[3] IMS LIP Specification, available at http://www.imsproject.org/

[4] McCalla, G. (1992). The central importance of student modelling to intelligent tutoring. In E. Costa (Ed.), New Directions for Intelligent Tutoring Systems. Berlin: Springer Verlag.

[5] McCalla, G. (1992). The central importance of student modelling to intelligent tutoring. In E. Costa (Ed.), New Directions for Intelligent Tutoring Systems. Berlin: Springer Verlag.

[6] Burton, R. (1982). Diagnosing bugs in a simple procedural skill. In D. Sleeman & L. Brown (Eds.), Intelligent Tutoring Systems. London: Academic Press.

[7] Burton, R. & Brown, J. (1978). Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-191.

[8] Clancey, W. (1986). Qualitative student models. Annual Review of Computer Science, 1, 381-450.

[9] Self, J. (1990). Bypassing the intractable problem of student modelling. In C. Frasson & G. Gauthier (Eds.), Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education. New Jersey: Ablex.

[10] Self, J. (1994). Formal approaches to student modelling. In G. McCalla & J. Greer (Eds.), Student Models: The Key to Individualized Educational Systems, New York. Springer Verlag.

[11] Corder, S. (1967). The significance of learners' errors. International Review of Applied Linguistics, 5, 161-170.

[12] Carr, B. & Goldstein, I. (1977). Overlays: A theory of modelling for computer-aided instruction. AI Lab Memo 406. Massachusetts Institute of Technology, Cambridge, Massachusetts.

[13] Sison, R. & Shimura M. (1998). Student Modelling and Machine Learning. International Journal of Artificial Intelligence in Education, 9, 128-158.

[14] Payne, S. & Squibb, H. (1990). Algebra malrules and cognitive accounts of errors. Cognitive Science, 14, 445-481.

[15] Baffes, P. & Mooney, R. (1996). Refinement-based student modelling and automated bug library construction. Journal of Artificial Intelligence in Education, 7(1), 75- 116.

[16] Newell A. and Simon H. (1972) Human Problem Solving. Englewood Cliffs, NJ. Prentice-Hall.

[17] Sleeman D. & Brown, J. S. (1982) Introduction: Intelligent Tutoring Systems. In D. Sleeman & J. S. Brown (Eds.), Intelligent Tutoring Systems. Academic Press

[18] Carbonell, J. (1970) AI in CAI: an Artificial-Intelligence approach to Computer-Assisted Instruction. IEEE Transactions on Man-Machine Systems, 11(4), 190-202

[19] Anderson, J. R. (1983). The Architecture of Cognition. Cambridge, Massachusetts: Harvard University Press.

[20] Koedinger, K. R., & Anderson, J. R. (1993). Reifying Implicit Planning in Geometry: Guidelines for Model-Based Intelligent Tutoring System Design. In S. P. Lajoie, Ed. & S. J. Derry, Ed (Eds.), Computers as Cognitive Tools (pp. 15-45). Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers.

[21] Reiser B. J., Anderson J. R., and Farrell R. G., "Dynamic student modeling in an intelligent tutor for Lisp programming," Proc. Ninth International Joint Conference on Artificial Intelligence, pp. 8--14, Los Angeles 1985.

[22] Beck, J. Stern, M., and Woolf , B. P. (1997) Using the Student Model to Control Problem Difficulty. In Anthony Jameson, Cécile Paris, and Carlo Tasso (Eds.), User Modeling: Proceedings of the Sixth International Conference, UM97. http://um.org.

[23] Kjeldsen, R. and Cohen, P. (1987). The evolution and performance of the GRANT system. IEEE

Expert, summer:73–79.

[24] Quillian, R. (1968). Semantic memory. In Minsky, M., editor, Semantic Information Proces-ing, pages 216–270. The MIT Press, Cambridge, MA, USA.

[25] Croft, W. and R.H.Thompson (1987) I3R: a new approach to the design of Document Retrieval Systems. Journal of the American Society for Information Science, 38(6):389–404.

[26] Croft, W., Lucia, T., Crigean, J., and Willet, P. (1989). Retrieving documents by plausible inference: an experimental study. Information Processing & Management, 25(6):599–614.

[27] Gurer, D., desJardins, M., and Schlager, M. (1995). Representing a students learning states and transitions. Presented at the 1995 American Association of Artificial Intelligence Spring Symposium on Representing Mental States and Mechanisms.

[28] Preece, S. (1981). A spreading activation model for information retrieval. PhD thesis, University of Illinois, Urbana-Champaign, USA

[29] Rumelhart, D. and Norman, D. (1983). Representation in memory. Technical report, Department of Psychology and Institute of Cognitive Science, UCSD La Jolla, USA.

[30] Sauers, R., and Farrell, R. (1982) GRAPES User's Manual. ONR Technical Report ONR-82-3, Carnegie-Mellon University.

[31] Anderson, J. and Reiser, B. (1985) The LISP Tutor. Byte, 10(4): 159-175

[32] Gagne, R. (1985). The Conditions of Learning (4th ed.). New York: Holt, Rinehart & Winston .

1.5 Overview

This Survey on Methods and Standards for Student Modelling contains the following information:

- Introduction. Introduces this document.
- Standard Modelling Techniques. This section discusses the theoretical aspects behind student modelling.
- Some Systems of Interest. This section shows some interesting research prototypes and discusses their peculiarities.
- Learning Standards. This section describes the main learning standards and some commercial systems that adopt them.

Throughout this document, terms like student, learner and trainee are used interchangeably. Though widely used in this document, the term "standard" when used for the current specification initiatives is slightly incorrect. In fact, at the moment there are no student modelling standards, but only specifications, that will need the final ratification of a sanctioning body to become official standards. Nevertheless, because of the wide (mis-)use of this term in literature, we will utilize it in this document too.

2 Standard Modelling Techniques

2.1 Introduction

Student modelling, as the model of a learner, represents the computer system's belief about the learner's knowledge. It is usually used as part of computer-based instructional systems and intelligent tutoring systems (ITS), existing within the wider domain of user modelling. It involves the construction of a (necessarily) qualitative representation that accounts for student behaviour regarding two aspects: existing background knowledge about a domain (1) and about students learning of the domain (2). See [4]. A third aspect (student observed behaviour) will be looked into later.

The student model is the essential component in individualized learning via an intelligent instructional system that is able to adapt to the learner. It is the student model that builds and maintains the system's understanding of the trainee. Intuitively speaking, the definition of a student model involves answering four basic questions:

- "who" is modelled: the level of detail and the approach in defining who and what the student history is;
- "what": the goals, plans, attitudes, capabilities, knowledge, and beliefs of the learner;
- "how" the model is to be acquired and maintained;
- "why"; that is, deciding whether to elicit information from, to give assistance to, or to provide feedback to the learner or, again, to interpret student behaviour.

These four intuitive questions can be reduced to three fundamental issues mentioned previously. Together they shape student modelling: (1) the student model itself (as the output of the overall process of student modelling), (2) background knowledge and (3) student behaviour.

We will examine these three aspects of student modelling theory in the following sections.

2.2 The Student Model

The first element of student modelling to take into consideration is the output of the student modelling process itself. A student model is an approximate, possibly partial, and primarily qualitative representation of student knowledge about a particular domain, or a particular topic or skill in that domain that can fully or partially account for specific aspects of student behaviour.

Student models are qualitative models. That is, they are neither numeric nor physical; rather, they describe objects and processes in terms of spatial, temporal, or causal relations [8]. Student models are approximate, possibly partial, and do not have to fully account for all aspects of student behaviour. In fact, we are interested in computational utility rather than in cognitive fidelity [9]. A more accurate or complete student model is not necessarily better, since the computational effort needed to improve accuracy or completeness may not be justified by the enhancement obtained, which may in practice prove to be only slight [10].

To recap, a student model is an approximate, possibly partial, qualitative representation of student knowledge about a particular domain that accounts for specific aspects of student behaviour. Accounting for behaviour, as we will see, involves identifying relationships between behaviour and background knowledge, and can be achieved at the behaviour, knowledge, and/or learning levels (these levels are examined below.) Constructing a student model which can deal with misconceptions as well as other knowledge-level errors can be achieved using either an analytic or a synthetic approach, or a combination of the two. One can of course think of other issues regarding intelligent tutoring or coaching, particularly issues relating to pedagogical activities, individual learning styles, etc. but these are beyond the scope of this document.

Several taxonomies have been proposed to categorize student models. One [5] is based on the different uses of a student model within an Intelligent Tutoring System. These are:

- *Corrective*. The student model is employed for discriminating discrepancies between "correct" knowledge and the student's current understanding. Other parts of the ITS will take charge of the necessary corrective actions.
- *Elaborative*. In this case the model is used for extending the student's knowledge by proposing new

topics or refining others, etc.

- *Strategic*. Taking a wider perspective, the student model can record "strategic" data about learners, like explicit representation of the teaching strategies currently adopted together with related performance ratings, etc.
- *Diagnostic*. The analysis of the state of the student by means of his or her model. In this case the tutor can infer some information about the learner by inspecting or querying opportunely the student model.
- *Predictive*. In this case the student model is used for anticipating the effect of an action upon the student. The model works like a simulator of the learner (limited to the sub-state of the learner that we are interested in).
- *Evaluative*. By means of the student model we can assess the learner's achievements.

Section 2.4 presents the different approaches to general student modelling.

2.3 User behaviour and the System Background Knowledge

Utilising a student model we can identify specific *relationships* between the input behaviour and the system's background knowledge. Such relationships can of course be analysed at various levels. Figure 1 shows the layering of such error levels, as discussed below.

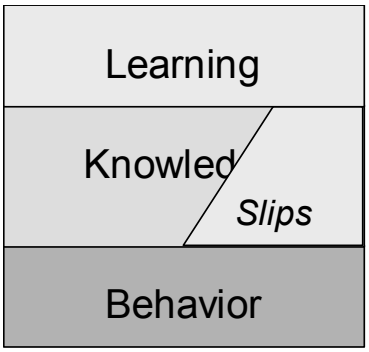

Figure 1. Level of Error in Student Modelling

At the most basic level, which in [13] is defined as the *Behaviour* level, the relationships between actual and desired behaviours (i.e., syntactic mismatches) are determined. These discrepancies are called behavioural or behaviour-level errors in that they occur through inappropriate actions performed by the learner. While the detection of behaviour-level errors is trivial when dealing with simple behaviours like writing numerical answers to math problems, it quickly becomes a nontrivial task when one begins to deal with more complex behaviours like writing lines of a computer program; here, to ascertain the acquisition of knowledge or detect a misunderstanding, the tutor should concentrate on more abstract patterns rather than on the mere syntactic validity of what the learner writes

Hence, at a higher level (which is known as the *Knowledge* level), the relationships, particularly causal relationships between behavioural discrepancies, become significant. It is at this level that misconceptions and other classes of knowledge errors are recognized. Misconceptions are incorrect or inconsistent beliefs, procedures, concepts, principles or strategies that result in behavioural errors. Not every error in behaviour is a result of incorrect or inconsistent knowledge, however, since behavioural errors can also be due to insufficient knowledge. We use the term "knowledge error" to include both incorrect or inconsistent knowledge (i.e. misconceptions) and missing or incomplete knowledge. Moreover, what could seem to have been caused by some knowledge error may actually be a contingent slip [11], due, for example, to fatigue, boredom or distraction. The errors at this level, both knowledge errors and slips, are called knowledge-level errors. At an even higher level – which we may call the "learning level", relationships between misconceptions and corresponding or analogous correct pieces or chunks of knowledge become important, as the former might derive from the latter. For example, a

misconception can be traced to an overgeneralization of an existing piece of knowledge. It is at this level that theories of (mis)learning such as the REPAIR theory become especially helpful. *REPAIR Theory* introduces a generalization of bugs, called "impasses". One example of a bug is dividing by zero when doing arithmetic; the impasse in the learner's conceptions concerns what to do when a denominator is zero. This lack of clear understanding may produce unpredictable results, often errors.

Each level of this hierarchy can be viewed as an explanation or generalization of the level below it; that is, knowledge-level errors explain the occurrence of errors at the superficial behaviour level, while learning-level errors explain the occurrence of misconceptions and other errors at the knowledge level. Higher-level knowledge is therefore more efficient to store, though more expensive to acquire. Student modelling systems are mainly concerned with errors at the knowledge level. It might not (always) be possible or useful to distinguish between knowledge errors and slips.

The knowledge base of the student model takes both domain and pedagogical knowledge into account.

2.4 Building Student Models

2.4.1 Expert-based modelling

In expert-based modelling, or the *overlay* approach [12], the student model is assumed to be a subset of the expert model. Most early systems used this approach (e.g. SCHOLAR, section 3.2). In this group of modelling approaches the domain representation of the tutoring system is thought to be a representation of some expert's knowledge, and the student model is built accordingly. First, the expert domain is modelled as a set of correct production rules (for example). The learner is modelled as a subset of these correct rules, plus a set of incorrect production rules. Each new learner requires an individualized student model. In developing the student model, the type of knowledge (i.e., declarative, procedural) to be defined must be determined. In addition, it has to be decided whether to include student goals, and how to include these. The LISP tutor is an example of this approach (section 3.4). The methods used include users defining their own goals, providing documentation about themselves, and/or submitting answers to a pre-test.

A number of assumptions stem from this approach.

- There is a definite (often explicit) objective to the learning. The "student model" is simply the representation of whatever knowledge the learner has, taken to be a *subset* of the expert's knowledge.
- Not only the knowledge, but also the *way* this is structured is the same both in the tutor and in the student.

Essentially, expert-based modelling methods can be divided into two main categories:

- Subset-based methods. In this case knowledge, thought to be "atomic" (built up out of elementary units of data), has been divided into discrete units, often hierarchically organized. There is no possibility of allowing the learner to have conceptions of the domain different from those of the expert. The atoms of the knowledge units cannot be rearranged in the student's model. Such a technique has been adopted for network representations and rule-based representations.
- Perturbation-based methods. With this approach the tutor is responsible for identifying and eliminating possible misconceptions as well as adding the correct information to the understanding of the student, as in the previous method.

The misconceptions in the student's understanding are called bugs. These are not errors, but higher-level misunderstandings, that generate many different errors. "16/0=0" or "2/0=2" are both errors, generated by the commonly held belief that divisions by zero are valid arithmetical operations: that belief is a misconception or bug. In the literature, "errors" are usually portrayed as behaviours, in that they appear through the learner's inappropriate actions, e.g. writing "0" after "2/0=".

2.4.2 Learner-based modelling

Expert-based approaches all rely on a common assumption, that the student is somehow a mini-expert. That is to say that the learner's deficiencies are only a lack of quantitative knowledge. There is sound psychological evidence that conceptions of a domain may radically alter over time. Many teaching approaches rely on this assumption.

Rather than focusing on the expert and forcing learners to become copies of the knowledge authority, learner-based approaches tend to focus on students, the ways in which they acquire knowledge, and describing what they acquire. While the expert-based approach necessarily precludes the diagnosis of misconceptions, i.e., incorrect (as opposed to missing) knowledge, the learner-based approach can be viewed as an attempt to deal with this limitation.

Two methods which adopt the latter approach are outlined here. The first uses background knowledge to transform student behaviour in handling the problem given (this presupposes the definition of desired outcomes of behaviour), or to verify if student behaviour and some desired behaviour are equivalent or not. The specific operators needed to perform the transformation make up a student model. This is called the *analytic* or *transformational* method. The other involves obtaining a set of behaviours and computing a generalization of these by, for example, synthesizing elements from the background knowledge or input data. The synthesized generalization makes up a student model. This is known as the *synthetic* method.

Normally, systems that need to be able to construct their student models from a single behaviour adopt an analytic method, while systems that construct their student models from multiple behaviours (like DEBUGGY, section 3.3) adopt a primarily synthetic method. The technique used in ASSERT (section 3.5), called theory revision, can be viewed as transformational, though it involves the transformation of a model rather than a behaviour. The transformation is guided by the data, i.e., by a set of behaviours, rather than by background knowledge. This means that the procedure used by ASSERT relies on multiple behaviours.

The Automated Cognitive Modelling, based on machine learning, constructs a student model off-line. Such a model is composed of several production rules that aim at mimicking student behaviour on a given set of problems. Generally speaking, learning is the induction of new, or the compilation of existing, knowledge; this, in turn, may lead to improvements in the performance of a task. Most machine learning research has focused on improving accuracy (and efficiency) in classification tasks, but research into improving efficiency (and accuracy) in problem solving has also been carried out.

2.5 Background Knowledge for Student Modelling

The background knowledge of a student modelling system usually comprises:

- The correct beliefs, procedures, concepts, principles and/or strategies of a domain (collectively called the *theory* of that domain).
- The misconceptions held and other errors made by a population of students in the same domain (collectively called the *bug library*).

The background knowledge may also contain historical knowledge about a particular student (like past qualitative models and quantitative measures of performance, student preferences and unconventional behaviours), and stereotypical knowledge about student populations in the given domain.

In actual practice, it is generally impossible to have a complete bug library. In effect, while it is sometimes possible for the theory of a given domain to be completely specified (for example, enumerating all the correct rules for arithmetic), it is difficult, if not impossible, to enumerate all the misconceptions that students may possibly have, even when one only considers those errors that students often tend to make. Furthermore, the results of [14] suggest that different groups or populations of students (e.g., students from different schools) may need different bug libraries. Hence, it is not a surprise that very few systems have the capability to automatically extend (let alone construct) their bug libraries.

2.6 Student Behaviour

[13] used the term *student behaviour* to refer to a student's observable response to a particular stimulus in a given domain. This, together with the stimulus, serves as the primary input to a student modelling system.

This input (i.e., the student behaviour) can be an action (e.g., solving a test) or, more commonly, the result of that action (for example the solution to the test). It can also include intermediate results (e.g., scratch work) and verbal protocols. In intelligent tutoring systems, stimuli from the tutor would typically come in the form of selected questions or problems about a particular topic.

Student modelling systems in the domain of concept learning (e.g. classification) and problem solving can be classified as to whether they can construct a student model from a single piece of behaviour or require multiple behaviours to accomplish their task. Currently, systems that deal with relatively simple behaviours such as integers (like DEBUGGY, see below), or propositions (like ASSERT [15]) require multiple behaviours to construct a student model. In fact, a student model that is inferred from a single item of simple behaviour such as an integer will generally not be reliable. Each of the systems just mentioned uses a machine learning or machine-learning-like technique such as rule induction (DEBUGGY), or theory revision (ASSERT) to synthesize a model that accounts for most, if not all, items in the input behaviour set. The granularity of the student behaviours utilised may also vary, for example, in the LISP tutor the student model is updated after the entry of a single LISP atom ('word') rather than waiting for a complete function or program to be entered.

To summarize, student behaviours, i.e. the observable responses of students (to domain stimuli) used (together with the stimuli) as the primary input for student modelling, can be simple or complex, and student models can be constructed from single or multiple behaviours. The background knowledge used to infer student models from behaviours may include a domain theory and/or a bug library. These are usually built manually; (ideally, they should be automatically extensible, even constructible, from scratch but this is currently beyond the reach of even state-of-the-art systems).

2.7 Domain Complexity

Another important aspect is how to tame the intrinsic complexity of the domain of interest – tutoring subtraction, for example, versus LISP programming. The notion of domain complexity can be described informally as follows (following [13]):

- Following Gagne's [32] hierarchy of intellectual skills, we can view problem solving tasks as more complex than concept learning or classification tasks in the sense that problem solving ability requires classification ability.
- Using the problem-space theory of human problem solving [16], we can regard some problem solving domains or tasks as more complex than others, depending on the complexity of state representations, how well the operators have been defined, and the nature of the search process (algorithmic vs. heuristic), among others. Complexity in problem solving domains can therefore be viewed as a spectrum, possibly multidimensional.

In maintaining the student model from a practical viewpoint, the fact that students, (1) do not perform consistently, (2) forget information randomly, and then (3) display large leaps in understanding, requires to be considered.

We conclude this section with a brief mention of the two main approaches to learning theory.

2.8 Two Schools of Thought

The creation of instructional material involves the organization of information to promote specific learning goals. To simplify, there are two schools of thought in the creation of learning instructions: objectivists and constructivists.

Objectivists believe they can define a series of steps that will lead the learner to the final goal. This final goal is defined in terms of behaviour, i.e. "The learner will be able to demonstrate behaviour x". This method does not take into account the individual learner's differences regarding prior knowledge or present motivation. It does expect a minimum prior skill relevant to the knowledge domain. This

approach may work for procedural knowledge (which can be exhibited) but is not as effective with declarative knowledge, and higher levels of learning. Moreover, although it may produce results in the long run, it is not necessarily the most efficient in economic terms (expenditure of student time and energy, prolonged use of equipment, etc). Nor is the final knowledge state necessarily extensible (the student may not be able to use creatively the knowledge acquired to extend its range of application).

The constructivist approach differs from the objectivist in that the student takes control of the learning process. One example of this approach is the Cognitive Flexibility Theory. This theory deals with the special requirements for attaining advanced learning goals. It views the learning methods as having a multi-dimensional perspective with a criss-crossing of the subject matter in a non-linear fashion. Constructivist learning does not have the disadvantages mentioned for objectivist learning; however it is not clear whether it is suitable for all domains and for all students, especially since the student's motivation to learn must be high.

3 Some Systems of Interest

3.1 Introduction

Some systems will now be outlined, from various fields, which are relevant to the student modelling task in Diogene. Table 1, below, provides a summary of the covered systems:

System	Year	Domain	Type of Student/User Model
SCHOLAR	1970	Geography of South America	Semantic network
DEBUGGY	1978	Subtraction	Procedural network
LISP Tutor	1985	LISP Programming	Production system (procedural rules, semantic network storing declarative knowledge)
GRANT	1987	Searching for a funding agency or research topic	Semantic network with spreading activation
I^3R	1987	General purpose information retrieval	Mixture of statistical (keyword) and semantic network
ASSERT	1990	C++ Programming	Feature vectors
Animal Watch	1997	Arithmetic	Student proficiency/belief vectors

Table 1: List of systems covered in this report

Each of these systems will be briefly described in turn.

3.2 SCHOLAR

SCHOLAR [18] is considered one of the earliest examples of the application of artificial intelligence techniques to Computer-Assisted Instruction (CAI – a term which was latter replaced by 'Intelligent Tutoring System', ITS). The system is designed to enable a student to engage in a question answering dialog with SCHOLAR, on the subject of the geography of South America. Figure 1 shows a short example (from [18]) of such a dialog. While the interface of SCHOLAR is perhaps dated, it showed the potential of using domain and user modelling to enrich the experience of a student, and introduced a number of important modelling techniques. In particular, the user modelling aspects are heavily influenced by the work of Quillian on semantic networks [24].

According to the Quillian's definition, Semantic Networks express knowledge in terms of concepts, their properties, and the hierarchical sub-superclass relationship between concepts. Each concept is represented by a node and the hierarchical relationship between concepts is depicted by connecting appropriate concept nodes via "is-a" or "instance-of" links. Nodes at the lowest level denote classes or categories of individuals while nodes at the higher levels denote classes or categories of individuals. Concepts get more abstract as one moves up the is-a hierarchy. Properties are also represented by nodes, and the fact that a property applies to a concept is represented by connecting the property node and the concept node via an appropriate labelled link. Typically, a property is attached at the highest concept in the conceptual hierarchy to which the property applies, and if a property is attached to a node, it is assumed that it applies to all nodes that are descendants of that node.

```
WHAT IS CHILE?

*A COUNTRY*

THAT'S FINE.


SELECT AN ALTERNATIVE FROM THE LIST:

214000.
800300.
49883.
63690.

FOR THE QUESTION:

    APPROX WHAT IS THE AREA OF CHILE?

*800300*
```

```
YOUR ANSWER IS INCORRECT.
THE CORRECT ANSWER IS:214000.


… etc …


*TELL ME SOMETHING ABOUT PERU.*

WAIT A MINUTE. I'M CHECKING MY FILES.
LET'S PROCEED NOW.

   PERU IS A COUNTRY.
   IT IS IN SOUTH AMERICA.
   THE POPULATION IS APPROX 7000000.
PEOPLE.
   THE CAPITAL IS LIMA.

… etc
```

Figure 1: Example student-SCHOLAR dialog (a short extract from [18]). Text entered by the student is surrounded by asterisks (e.g. "*A COUNTRY*"), all other text is generated by SCHOLAR.

The semantic network with which SCHOLAR operates is intended to model a geographic area (in this case, South America). The idealised model represented by the semantic network is taken as a model of an 'ideal' student – a student interacting with SCHOLAR is then considered as a subset of this ideal model: "… the modelling of a student is made much easier by giving him the benefit of the doubt and assuming he is correct until proven wrong". An example of the semantic network used is shown in Figure 2. Such a network is composed of individual *units* (the boxes in Figure 2), each unit being composed of a set of *properties*, the first property normally being the name of the unit (for example, 'ARGENTINA'). Each property is made up of a name, set of tags, and value. Each value can 'point' to other units in the network. In Figure 2, for example, the value 'URUGUAY' points to the unit 'URUGUAY'. While this representation is largely declarative in nature, procedural code can also be inserted to implement reasoning on units. Context within this structure is also encoded by the creator of the network,, who is able to tag values based on their relevance to the unit in question.

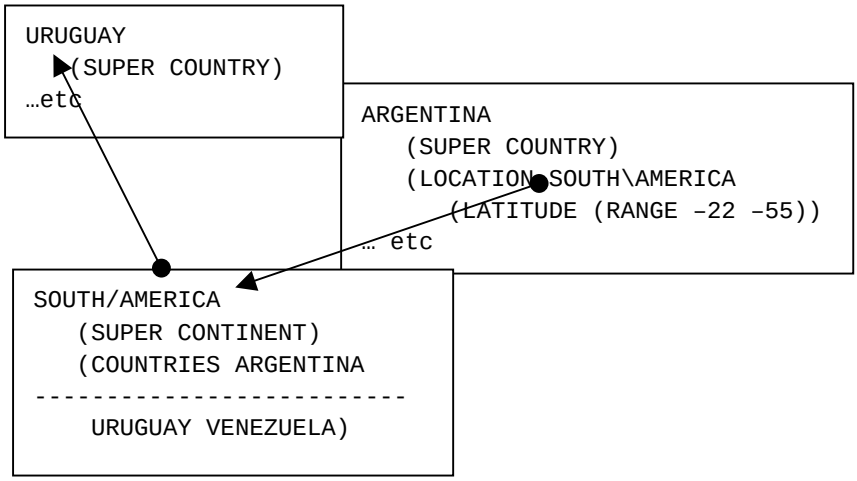

Figure 2: A small extract from the SCHOLAR semantic network (from [18])

The process of evaluating the student's performance in the task is carried out by generating the 'right' answers from the semantic network, and comparing this answer with the one given by the student. A process of error analysis then attempts to separate the aspect of the students answer which was correct

from that which was incorrect, to produce a diagnosis, for which a taxonomy of errors was developed. Some classes of errors listed in [18] include:

- Missing information (not knowing a fact)
- Misfiled fact (e.g. putting a city in the wrong country)
- Lack of a concept (e.g. not knowing what longitude is)
- Overgeneralisation error (e.g. all governments are military)
- Failure to draw a superordinate inference (e.g. the language of a country implies, normally, the language of the cities within that country)
- Failure to draw a negative inference (e.g. failure to recognise a contradiction)

No quantitative or qualitative evaluation of SCHOLAR with real users is given in [18], although the system is well read in the ITS domain, if only for historical reasons.

3.3 DEBUGGY

DEBUGGY [6] is a diagnostic system that extends an earlier BUGGY model [7], in which a skill such as place-value subtraction (BUGGY/DEBUGGY's domain) is represented as a procedural network (that is, a recursive decomposition of a skill into subskills/subprocedures). A student modelling scheme based on such a network requires that the background knowledge contains all the necessary subskills for the general skill, as well as all the possible incorrect variants of each subskill. A student modelling system can then replace one or more subskills in the procedural network by one of their respective incorrect variants, in an attempt to reproduce a student's incorrect behaviour. The procedure is as follows:

1. From a set of (130) predefined buggy operators, select those that explain at least one wrong answer in the student's behaviour set, B. Call this initial hypothesis set H.
2. Reduce H by removing buggy operators that are subsumed by others, giving H'.
3. Compound every pair of buggy operators in H' to see if the resulting bug covers more answers than either of its constituents. If so, add this compound to H'. Repeat this step for each of the new compounds, giving H''.
4. From H'', select a set of bugs, P, that explain a given percentage of the student's answers. For every bug in P, first identify the student answers for which the bug predicts a different answer, then coerce (using heuristic perturbation operators) the bug so that it reproduces the student's behaviour.
5. Classify and rank the bugs in P according to the number of predicted correct and incorrect answers, the number and type of mispredictions, and the number and type of coercions. Choose the one with the highest score.

The incorrect variants of skills/subskills are called bugs in the BUGGY framework. Note that a bug in this approach denotes a knowledge error (e.g., a misconception), rather than a behavioural one. Thus, it is not to be confused with programming bugs, which refer to errors in a program that, in the context of student modelling, are errors at the behavioural (rather than at the cognitive) level. Furthermore, in the terminology of [6] the term *bug* is sometimes used synonymously with what could be called a buggy model, i.e., the procedural network that results when the correct sub-procedure that corresponds to a particular bug is replaced with that bug. This synonymous treatment is due to the fact that bugs, rather than buggy models, are what DEBUGGY manipulates explicitly; a buggy model can always be generated at any time for any bug. A student model in DEBUGGY is therefore an individualized procedural network.

To construct the student model, a simpler approach could have simply generated a set of buggy models, one model for each primitive bug in the bug library, selecting the model that reproduces all or most of a student's answers to a set of problems. This wouldn't be practically feasible, because the search space when compound (i.e. multiple) bugs are responsible for the student's answers is too big. Instead, DEBUGGY assumes that the primitive bugs that interact to form a compound bug are individually detectable, so that there is at least one problem in which they appear in isolation.

Thus, to determine a student's compound bug (and, in the process, the student's buggy model) given the

student's behaviour set, DEBUGGY hypothesizes, from 110 predefined primitive bugs and 20 compound bugs, an initial set of bugs whose elements it then removes, compounds, or coerces. The elements of the final set are then ranked, and the bug with the highest score is outputted. From this procedure, we can see that DEBUGGY actually performs a kind of supervised inductive learning of a student model, or more exactly, of a compound bug.

3.4 LISP Tutor

LISP tutor [21, 31], also known as GREATERP, is an Intelligent Tutoring System developed to teach the basic principles of programming in LISP. The system is based on Anderson's ACT* theory [19], which uses a production system as a model for human cognition (the theory being more general than the learning domain). The main principles of the ACT* theory are:

- Cognitive functions can be represented as a set of production rules. The use of a production depends on the state of the system and the current goals.
- Knowledge is learned declaratively through instructions. The learner must carry out the process of knowledge compilation if the productions are to be properly understood and integrated into their existing knowledge and later recalled and used.

The GRAPES, Goal Restricted Production System Architecture [30], is used to represent the knowledge in LISP tutor, and therefore combines declarative knowledge in the form of semantic nets with procedural knowledge in the form of the production rules. In ACT* learning is accomplished by forming new procedures through the combination of existing production rules.

The tutor is intended to embody a number of pedagogical objectives:

- the tutor should provide a complete, friendly environment for instructing the student
- students should do as much of the work as possible
- the tutor should provide immediate feedback
- the tutor should represent the structure of the problem the student is trying to solve

A "Model-Tracing" methodology is used, where each element of user action is tracked, and feedback is immediately provided to the user when an illegal or wrong piece of program code is entered. This requires three main components:

- An Ideal Student Model – a set of rules which produce correct LISP programs
- Bug Catalogue – a set of rules encoding common mistakes and errors
- Tutoring Control Module – pedagogical strategies which structure the interaction with the student

The three elements together are the generic student model, common to all students. This model is personalised to individual students by overlaying this structure with a set of weights – each weight encodes the degree of knowledge the student is thought to have of the particular rule. An example production rule (in structured English) is shown below, taken from [21]:

```
   IF the goal is to combine LIST1 and LIST2 into a single list
      and LIST1 is a list
      and LIST2 is a list
THEN use the function APPEND
     and set the subgoals to code LIST1 and LIST2
```

The 'if' part of the rule supplies conditions which are evaluated to determine if the rule applies. If the rule does apply, the 'then' part is executed – in the case above two new subgoals are created which will in turn be evaluated by other rules in the system. There are 375 correct and 475 buggy production rules, of the above form, in the LISP tutor.

When the next element (or 'atom') of the LISP program is entered by the student, it is evaluated against the production rules in the system (both correct and buggy). If one of the correct rules covers the

student's action, the student is not in error, and is allowed to continue normally. If a buggy rule covers the observed behaviour, the tutor will immediately provide error information to the student, to enable them to correct their mistake as soon as possible. An example screen showing this situation (taken from [21]) is shown in Figure 3. Also shown on this figure are the code place markers (such as '<REPEAT>'), used by the editor to indicate the code which can be validly entered at that point in the program.

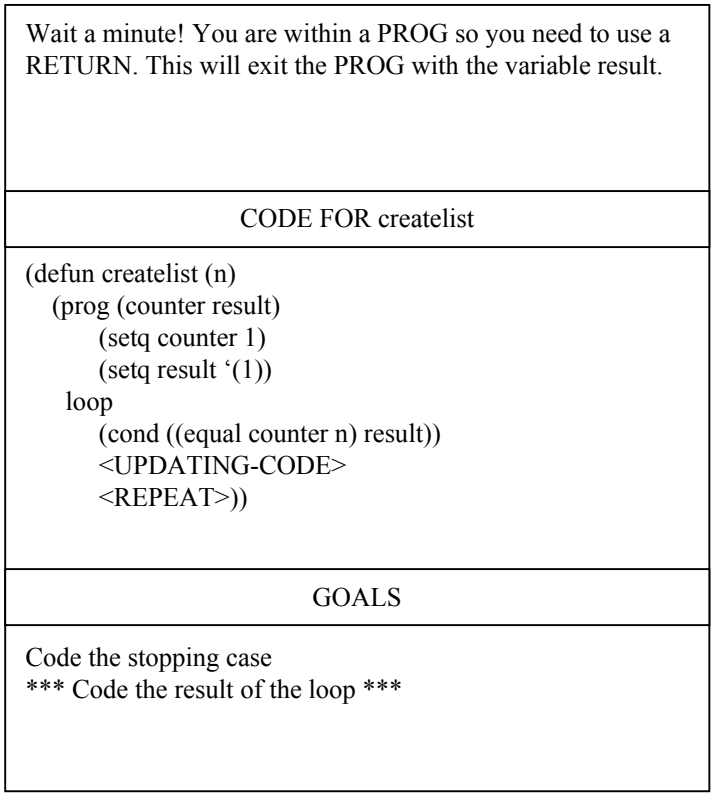

Figure 3: Example LIST tutor dialogue with the learner (from [21])

By providing immediate feedback to the learner, the student model the system builds can be simpler since the number of potential bugs at each step of the interaction is minimised. Forcing the learner to keep to a correct solution is one of the aims of the system, although this can be viewed as unnecessarily restrictive and counter-productive as the student is never allowed to explore incorrect behaviour. In common with other systems, much effort has to be put into developing the production rules – it is reported in [21] that between 45% and 80% of the errors a student makes can be correctly identified by the system depending on the complexity of the task and the quantity of testing put into developing the rules (newly developed rules only identified 45% of errors). Evaluation of the system [21, 31] with learner has found the system to be almost as effective as a personal tutor, and much more effective than classroom education.

3.5 ASSERT

ASSERT models students performing the classification of C++ programs according to the type of errors the programs may have, like for example the "attempt to assign a value to a constant" type of errors. ASSERT deals with propositions as behaviours (whereas DEBUGGY deals with integers, as we saw). It employs a machine learning technique, known as Theory Revision, which can be viewed as transformational, because it involves the transformation of the model, rather than the behaviour. ASSERT models student classifications of programs represented as feature vectors.

Unlike DEBUGGY, which synthesizes student models from primitives, ASSERT [14] immediately starts with a model, although a correct model (see step 1 below), which it then transforms to one that

covers the items in a student's behaviour set (see step 2 below). Thus, unlike the DEBUGGY approach which is synthetic, this approach is transformational. Yet, like DEBUGGY, it is inductive, since the transformation is nevertheless guided by the data. This transformational yet inductive approach to supervised learning is called theory revision or theory refinement.

The basic procedure used by the ASSERT system is as follows:

1. Initialise the student model to the ideal model.
2. Revise the student model by iterating through the following three steps until all student answers in the behaviour set are covered:
   - 2.1. Find a student answer in the behaviour set that is either covered by the current model when it shouldn't be (i.e. a false positive), or not covered by the current model when it should be (i.e. a false negative). Find a revision (delete rule, delete condition) for this falsely covered/uncovered example.
   - 2.2. Test the revision against all the other answers in the behaviour set. If the entire behaviour set is covered, apply the revision to the model.
   - 2.3. If the behaviour set is not covered entirely, induce new rules/conditions using an inductive rule learner.

The reader familiar with machine learning may already have noticed that ASSERT adopts an interesting variant of the basic theory revision setup. Whereas the basic setup transforms an incorrect theory so that it covers the training examples correctly and consistently, ASSERT transforms a correct theory (i.e. the correct model) so that it covers the student's set of possibly incorrect behaviour. Incidentally, THEMIS also begins with a correct model, though this choice was more pragmatic (less questions to ask the student) rather than paradigmatic. Step 2 of the procedure outlined above is carried out by the NEITHER propositional theory revision system, developed by the same authors.

Each student behaviour that ASSERT takes as input comes in the form of a pair, `<f; l>`, where `f` is a list of attribute-value pairs that describe an instance, and `l` is a label denoting what the student believes to be the class of the instance. An attribute-value pair, `<ai ; vj >`, is composed of a predefined attribute, `ai`, and the value, `vj`, of that attribute for a particular instance. Unlike DEBUGGY which models students performing problem solving (subtraction) tasks, ASSERT models students performing classification tasks, specifically, the classification of C++ programs (represented as feature vectors) according to the type of program error the programs may have, e.g., constant-not-initialised, constant-assigned-a-value.

We will see now how the ASSERT system constructs the Bug Library. We have seen that ASSERT can construct a student model without the need for any library of primitive buggy rules in the background knowledge. It is in the process of constructing a model that the system learns new buggy rules. By storing these buggy rules (in a principled way) into the background knowledge for later reuse, ASSERT is therefore capable of constructing its bug library from scratch.

ASSERT constructs a student model by refining the correct model (deleting a rule, deleting a condition, etc.). The set of refinements in a student model are collectively called a bug (cf. DEBUGGY's compound bug). ASSERT does not just store every bug in the bug library, however. Rather, it stores the most 'stereotypical' generalization of each bug. To do this, ASSERT first extracts the bug of each (buggy) student model and then determines the frequency, or what its developers call "stereotypicality", of each bug using the following formula:

$$S(B_i) = \Sigma_{j=1}^{n} \text{ distance } (C, M_j) - \Sigma_{j=1}^{n} \text{ distance } (M_i, M_j)$$

Where:

- M_i is the model,
- B_i are the refinements in model M_i,
- C is the correct model,

– n is the number of (buggy) student models.

It then tries to determine the generalization of each bug (i.e. the intersection of this bug, or its generalization in a previous iteration, with every bug other than itself) that has the highest stereotypicality value. This generalized bug, rather than the original bug (though the generalized bug might be the same as the original bug) is what ASSERT adds to the bug library.

The procedure for building a bug library in ASSERT is as follows.

1. Collect the refinements (i.e. bugs) of each student model into a list, B, and remove duplicates. Compute the stereotypicality of each bug, Bi, in B.
2. For each bug, B_i, in B:

2.1 Compute the intersection between B_i and every bug in B, and determine the intersection Ii with the highest stereotypicality. If the stereotypicality of I_i exceeds that of B_i , redo this step using Ii in place of B_i .

2.2 Add I_i to the bug library.

3. Rank the bugs in the bug library according to stereotypicality.

3.6 AnimalWatch

Animalwatch tutors addition, subtraction, multiplication and division of whole numbers and fractions, utilising word problems and the metaphor of endangered species in order to appeal to younger students. Animalwatch is intelligent in that it provides students with problems for which the student is "ready". For example, a student is ready to see a word problem that involves fractions only after she has shown proficiency in all the topics that involve whole numbers. Also, there is a notion of difficulty of a problem within a topic that depends on other factors (size of the operators, number of steps involved in solving the problem, etc.). The student model is used for topic selection, problem generation and hint selection.

Each type of problem in AnimalWatch is called a ‘topic’. Each topic has associated with it ‘pretopics’ (other topics which must be understood by a student before this topic can be given) and ‘subskills’ (the steps in the topic’s problem solving process).

The student model records a *proficiency* for each topic within the domain, and also some general student ability factors. Each proficiency is actually a history of proficiency scores over time, allowing the system to record the ups and downs of a student’s activity on a particular topic. A simple number for the proficiency score, such as 0.4 (on a 0 to 1 scale) does not provide information about the context of the student’s performance. Is the proficiency 0.4 because the student has just started on the topic, but is doing well? Or is the student having difficulty and his proficiency used to be higher? A history-based model is used, so that the tutor can track the student’s performance over time. This history is used in the selection of high-level teaching strategies such as whether the student needs remediation or if he has forgotten a topic and needs a review.

Each proficiency score is a belief vector, allowing the system to express the degree of confidence in the ability of a student at different ability levels. Each vector contains seven points, with the values summing to 1. The value at each point indicates the approximate probability that the student is at that level of knowledge. The lowest value for the vector is (1 0 0 0 0 0 0) and the highest value is (0 0 0 0 0 0 1). A vector of (0.14 0.28 0.4 0.18 0 0 0) means the tutor believes there is approximately a 14% chance the student is at level 1, a 28% chance he is at level 2, a 40% chance he is at level 3, and an 18% chance he is at level 4. There is no chance he is above level 4. Maintaining a list of these vectors enables the tutor to evaluate the student in the context of his past work. This representation does not actually use probability theory, but is updated via heuristics.

In addition to the proficiencies of a student on particular topics, the system also records some general attributes about the student:

- Acquisition, which is a gauge of the how well new topics are being learned by the student.
- Retention, which is a gauge of how well the student remembers the material being presented.

Such general student characteristics are beneficial for student modelling. The student model is used in

the three ways previously mentioned:

- Topic selection: topics are initially selected on the basis of the proficiencies, acquisition and retention values, and topics which may, for example, have been forgotten by the student are first selected. If no such topic exists, heuristics are used to select among the topics still to be mastered.
- Hint selection: during the course of problem solving, Animalwatch can provide hints to the student. Each hint is of a particular level corresponding to the amount of information provided to the student. Heuristics are used in this particular system to determine what hint to provide to a student (for example, if they have gone wrong) and to diagnose where a student went wrong.
- Problem selection: this is the most important of the processes to Animalwatch, given the emphasis on the tailoring of the instruction to an individual.

On the subject of problem selection, from [22]:

> "The general philosophy is that the more subskills required to solve a problem, the harder the problem … the goal of the tutor in generating problems is to determine how many and which subskills are needed when solving the problem … For each problem generated, the tutor dynamically makes these decisions, and produces a problem to fit the chosen criteria."

The two main tasks are therefore to find:

- The number of subskills required, which is based on the ability and acquisition level of the student. The better the student is performing using the material, and/or the better a student is remembering the material, the harder the problems should be.
- The particular subskills to select. Subskills are selected based on the student's proficiency on the subtopics – the lower the proficiency, the greater the priority in presenting that topic to the student. In addition, subskills also have a proficiency level which relates to the difficulty of the subskill in question (the higher the proficiency, the harder the problem). Randomness is built into the selection procedure to provide an element of challenge or review for the student.

To update the student model, upgrade and downgrade rules, based on [27], which update the values in the belief vectors. Extending the work in [27], AnimalWatch attempts to better shift the belief vector to what the system infers to be the student's level of ability. This is achieved by tracking the levels of the hints supplied to a student, the upgrade and downgrade formulas being applied together to different parts of the belief vectors based on the highest level hint (the 'biggest' hint) given to the student.

3.7 GRANT

P.R. Cohen and R. Kjeldsen's GRANT system aims to "find funding sources that are likely to fund a given research project" [23], at heart a matching problem. Knowledge about research proposals and potential funding agencies is organised using a semantic network. Research topics and agencies are connected using a wide variety of association links to form a dense network. A query expresses one or more research topics, or one or more funding agencies. The search is carried out by constrained spreading activation on a semantic network.

Spreading Activation is based on supposed mechanisms of human memory operations, originating from psychological studies (see for example [29]). It was first introduced into Computing Science in the area of Artificial Intelligence to provide a processing framework for semantic networks. Its use has been praised and criticised, but it is currently adopted in many different areas including student modelling (e.g. SCHOLAR, 3.2).

Each node of the network is represented as a 'frame', which can be of ten different classes (from [23]):

Design	Educate	Improve	Intervene	Manage
Supply	Promote	Protect	Study	Train

Each of these 'case frames' has a set of optional 'slots' which are used to identify particular attributes of the nodes (for example, a study frame, with subject slot which identifies the topic of study). The processing technique used in spreading activation is defined by a sequence of iterations like the one schematically described in Figure 4.

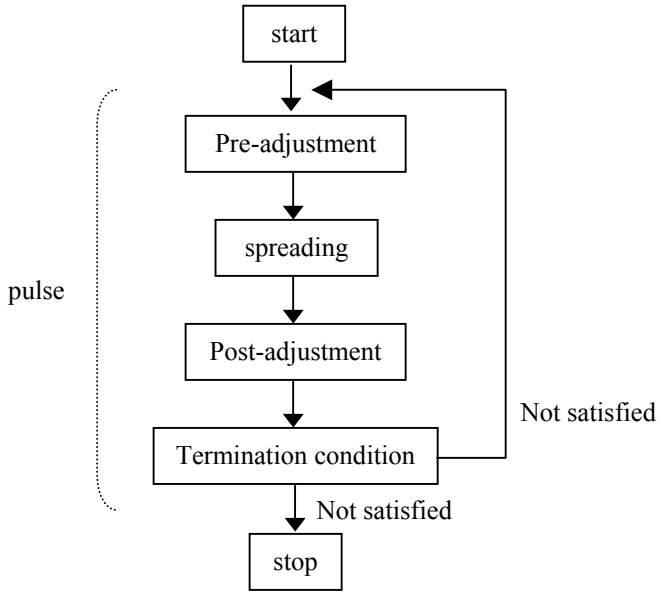

Figure 4: Overview of the steps in the pure spreading activation model, over a semantic network

Each iteration is followed by another iteration until halted by the user or by the triggering of some termination condition. An iteration consists of one or more pulses followed by a termination check. What distinguishes the pure spreading activation model from other more complex models is the sequence of actions which composes the pulse. A pulse is made up of three phases:

1. Preadjustment;
2. Spreading;
3. Postadjustment.

In the preadjustment and postadjustement phases, which are optional, some form of activation decay can be applied to the active nodes. These phases are used to avoid retention of activation from previous pulses, enabling the control of both activation of single nodes and the overall activation of the network. They implement a form of "loss of interest" in nodes that are not continually activated. The spreading phase consists on a number of passages of activation weaves from one node to all other nodes connected to it. There are many ways of spreading the activation over a network (for a overview see [28]).

Four heuristics are used to constrain the activation through the network: distance constraint, the spreading of activation ceases when it reaches nodes that are far away in terms of links covered to reach them from the initially activated ones; fan-out constraint**,** the spreading of activation should cease at nodes with very high connectivity, or fan-out, that is at nodes connected to a very large number of other nodes; path constraint**,** activation should spread using preferential paths, reflecting application dependent inference rules; and activation constraint**:** using the threshold function at a single node level, it is possible to control the spreading of the activation on the network, achieved by changing the threshold value in relation to the total level of activation over the entire network. GRANT uses a combination of all of there techniques, in particular path constraints in the form of 'path endorsement'.

From an heuristic point of view, GRANT can be considered as an inference system that applies repeatedly a single inference schema:

$$IF\ x\ AND\ R(x,y) \rightarrow y$$

where $R(x,y)$ is a path connecting the two nodes and which could be of one or more links. In the particular application for which GRANT was designed, i.e. finding founding agencies for research proposals, this is equivalent to an inference rule of the form: "If a founding agency is interested in topic

x and there is a relation between topic *x* and topic *y*than the founding agency is likely to be interested in the related topic *y*". The path endorsement process gives preference (positive endorsement) to some paths and it enables the avoidance (with a negative endorsement) of some misleading paths. The evaluation mechanism of the paths enables the retrieved nodes to be ranked.

The use of constraints on the spreading of the activation over the network and of rules to "endorse" some particular paths enable the system to achieve very interesting results - found to be better than those provided by simple keyword searches. This technique has been demonstrated to be particularly good for "difficult cases", that is, for cases that could have been difficult even for a human expert, though sometimes it provided misleading results for "simple cases". Developing a system like GRANT involves, first of all, a significant amount of knowledge engineering to construct the Semantic Network. This work consists of an in-depth analysis of the domain in which the system will operate in order to determine the appropriate concepts and relationships to build in the network, and the preferences to give to paths of activation spreading over it.

3.8 I^3R

I^3R is designed to act as a search intermediary, providing a user with a range of facilities for query formulation, browsing, retrieval, domain knowledge acquisition, and evaluation. It accomplishes its task using domain knowledge to refine query descriptions, initiating the appropriate search strategies, assisting the users in evaluating the output, and reformulating queries. A three-step process is used (from [26]):

1. Construct a representation of the information request
2. Retrieve documents using this request
3. Evaluate the results and reformulate the request

The system is organised around 'knowledge sources' which are used at various stages of the search process, some of which are listed below:

- User model builder
- Request model builder
- Domain knowledge expert
- Search controller
- Browsing expert

In terms of the user model, the two most interesting modules are the 'user model builder', which attempts to classify the user as belonging to a number of stereotypes, and the 'request model builder', which constructs a model of the user's information need (based on the request). The stereotypes in I^3R define the goals, the domain the user is interested in, and the style of interaction (among other things) of the user, and can be thought of, loosely, as acting in a similar manner to the expect models in a system such as SCHOLAR.

The authors implemented a retrieval paradigm called "multiple sources of evidence" - this paradigm being the central point of the research using I^3R, mixing various statistical and 'semantic' approaches. It springs from the intuition that a document is more likely to be relevant if its relevance is supported by many different clues. The use of multiple sources of evidence may potentially have some relevance to Diogene, where there exists the potential of using a combination of keyword and ontology approaches.

In its initial version [25] the domain knowledge was represented using an AND/OR tree of concept frames, while documents were represented by means of single term descriptors. In latter versions of I^3R the representation structure was refined to a sort of Semantic Network, of the kind depicted in Figure 5. Looking at that figure, we must remember that it is not necessary to distinguish among concepts, terms, and documents in the network structure. Importantly, much of this structure is automatically generated though the use of statistical techniques such as document-document similarity measures, natural language processing techniques (which the authors of I^3R started to utilise in later systems) as well as using predefined semantic information, such as thesauri.

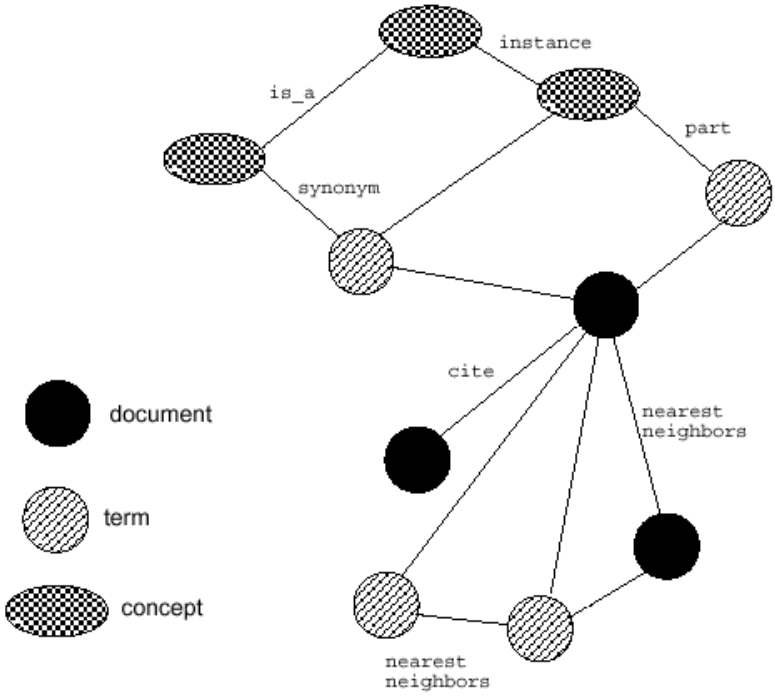

Figure 5: I3R network representation structure

Several processing techniques have been used on such a representation structure that could also be used for browsing. In particular in [26] the following specific form of constrained spreading activation was used:

1. The starting points of the spreading activation are the top-ranked documents from a probabilistic search;
2. Initially links connecting a document's nearest neighbours and document citations are used for spreading activation; these links represent the strongest plausible relationships between documents;
3. In the remaining cycles of activation only nearest neighbours' links are used; citation relationships are interesting only in relation to the starting documents;
4. Weights on links are used in the evaluation of the node's activation level; they are specified as "credibility" values associated to inference rules representing the existence relationships between the two nodes;
5. Documents that have been used as part of an activation path are not used again if they are reactivated.

3.9 A Brief Discussion of the Presented Systems

All of the systems described above have the capability to inductively construct user models that are consistent with a majority of the items in the behaviour sets of their users. Understandably, older systems such as SCHOLAR or DEBUGGY are less efficient in accomplishing this than newer ones such as ASSERT. However, the older systems did tend to deal directly with problem solving tasks (e.g., subtraction), whereas ASSERT deals with concept learning tasks, although one can, just by simplifying assumptions, reformulate problem solving tasks such as subtraction into classification tasks.

The ability to cover the data is, however, not the only concern for empirical learners who work with real-world data (as opposed to artificial data). Dealing with real data means dealing with imperfect, *noisy* data. Noisy data is the norm rather than the exception in student modelling, which will only be exacerbated in Diogene given the requirement to deal with unstructured textual data.

In literature, the phenomenon of wrongly covering noisy data (which should not be covered in the first

place) is called *overfitting*. To avoid overfitting of noisy data many machine learning systems either use stopping criteria which allow them to preterminate induction before all examples are covered, or review their output at the end of the induction process in order to delete or otherwise modify certain components according to certain criteria. There is still no general solution to the problem of noise, however, and not all research prototypes deal with this issue.

DEBUGGY addresses noise, particularly noise due to slips, via its coercion operator. However, this technique is somewhat limited, because slips cannot account for all the noise. SCHOLAR and ASSERT do not deal explicitly with noise. I^3R's approach of utilising multiple sources of evidence, including statistical keyword information, may be beneficial here, the assumption being that multiple sources of evidence will overlap allowing the common data to be more reliably utilised by a system. GRANT also uses a mixture of keyword and semantic properties in its matching algorithm, which would suggest a greater tolerance to noise than a purely 'semantic' matching on properties.

Finally, most supervised inductive machine learning systems require fairly complete background knowledge in the form of features and relations. SCHOLAR, DEBUGGY, ASSERT, GRANT, AnimalWatch and Lisp tutor all require detailed networks or production systems to be constructed of the domain being covered. I^3R, however, can use statistical, domain-independent techniques in its searching, providing some domain independence. In student modelling the completeness of conditions and operators (correct as well as incorrect) simply cannot be guaranteed, especially in more complex problem solving domains. In supervised inductive machine learning research, this problem is dealt with to some extent by constructive induction. DEBUGGY's technique of learning compound buggy operators can be viewed as a form of this. However, in addition to the efficiency issues surrounding this approach, DEBUGGY does not remember these buggy operators (and therefore has to recompute them every time) and, even if it did, a principled method of adding these to the background knowledge is itself a separate major issue. Furthermore, it is not clear how this technique can be used in a reasonable manner in problem solving domains that are more complex than subtraction.

Interestingly, many of the systems utilise semantic networks in their modelling process. Since their introduction by Quillian [24], *Semantic Networks* have played a significant role in knowledge representation research. All the systems described here take a different approach in their use of such networks, extending and restricting the definition as required for the task at hand (for example, SCHOLAR vs. GRANT).

While many of the systems (such as SCHOLAR, DEBUGGY and AnimalWatch) are ad-hoc in their approach to constructing user models, some systems, in particular the Lisp tutor, have been based on more general theories of human cognition. Underlying Lisp tutor is ACT* [19]. This is reflected in the aims of Lisp tutor, which are not only to provide an effective Intelligent Tutoring System, but also to provide a test bed of the ACT* theory.

GRANT and I^3R provide interesting examples of the mixing of keyword and semantic based techniques for information retrieval, and I^3R also provides an example of an information retrieval system which attempts to build a model of the user's searching. Initial results using GRANT showed its semantic-based matching to be superior to keyword based approaches, although in latter evaluations the difference is much less pronounced [23]. I^3R's mixture of methods provides a ready source of ideas for developing user-oriented systems mixing traditional and semantic information systems, in particular its use of multiple sources of evidence.

We have examined a number of interesting research systems and discussed some aspects of prototypical systems in this field, and in the wider user modelling field. We will now move to the main student model guidelines and specifications.

4 Learning Standards

The e-learning standard definition landscape is quite a complex one. Currently, half a dozen organizations are working to develop industry standards in this field. They include the following:

- IMS, a consortium of members that include major software developers and vendors, training and education representatives, and government agencies.
- ADL, an initiative of the U.S. Department of Defence which aims to ensure the interoperability of future e-learning technologies purchased by the U.S. government.
- LTSC, a branch of the Institute of Electrical and Electronic Engineers (IEEE) with a long-standing reputation as an accrediting body for technology standards.
- AICC (the Aviation Industry CBT Committee), an association of technology developers that spans well beyond the aviation industry.

All these groups are increasingly working together in order to harmonize their efforts towards a comprehensive set of standards for distance learning.

4.1 An Introduction to Learner Model Standards

The theory and research behind sophisticated learner models has been described above. However, aims and approaches change considerably where practical standards aimed at commercial learning systems are concerned. This section will examine the major student modelling specifications.

4.2 IEEE P1484.2 (PAPI Learner)

Public and Private Information (PAPI) for Learners (PAPI Learner) is a standard effort aimed at providing the syntax and semantics of a student model, including knowledge, learning styles, skills, abilities, records and personal information, all at multiple levels of granularity. This standard specifies the syntax and semantics of a "Learner Model", which characterizes a learner (either a student or knowledge worker[1]) and his or her knowledge/abilities. This will include elements such as knowledge (from coarse to fine-grained), skills, abilities, learning styles, records, and personal information. The standard will allow these elements to be represented in multiple levels of granularity, from a coarse overview, down to the smallest conceivable sub-element. It will allow different *views* of the Learner Model (learner, teacher, parent, school, employer, etc.) while addressing the sensitive issues of privacy and security.

The working group for the Learner Model [P1848.2] has the following purposes:

- To enable learners (students or knowledge workers) to build lifelong personal learner models.
- To enable personalized instruction and effective instruction.
- To provide educational researchers with a standardized source of data.
- To provide a foundation for the development of additional educational standards, from a student-centred learning focus.
- To provide architectural guidance to developers of education systems.

[1] In the meaning discussed in the "Motivations" section in the Task 1 document.

–

A simple view of the IEEE 1484.2 overall organization is provided in Figure 2.

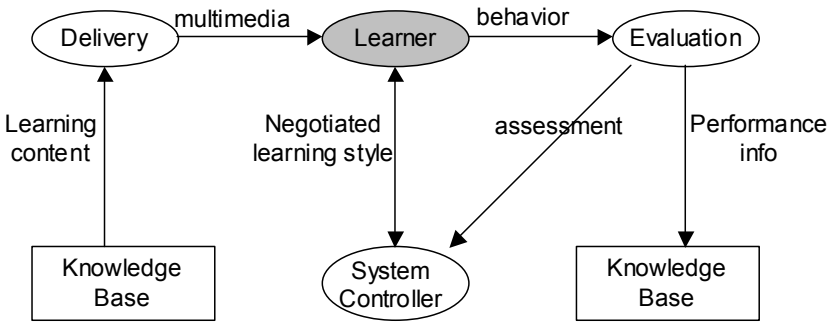

Figure 2. The main structure of Standard IEEE1484.2

4.2.1 Structure

The main architectural feature of the PAPI Learner standard is its logical division. It separates the security and the administration of several types of learner information (also called *Profile Information* or *Learner Profiles*):

- *Personal information* like name, address and social security number. It is not directly related to the measurement and recording of learner performance and is primarily concerned with administration. Usually this type of information is private and secure.
- *Relations information*, e.g., cohorts, classmates. This concerns the learner's relationship to other users of learning technology systems, such as teachers, practitioners, and other learners.
- *Security information*. This is concerned with the learner's security credentials, such as passwords, challenges/responses, private and public cryptographic keys, and biometrics.
- *Preference information*: useful and unusable I/O devices, learning styles and physical limitations. It describes preferences that may improve human-computer interactions.
- *Performance information*, like grades, interim reports, log books. This pertains to the learner's history, current work or future objectives and is created and used by learning technology components to supply enhanced learning experiences.
- *Portfolio information*: accomplishments, works and so on. This information is a representative collection of a learner's works or references to them that is intended to illustrate and justify the student's abilities and attainments.

The PAPI Learner Standard is not limited to these six types of information and may be integrated with other systems, protocols, formats, and technologies.

4.2.2 Overall Approach

Standards that attempt to be omni-comprehensive risk becoming large and unmanageable. The designers of PAPI Learner aimed at concrete diffusion by avoiding catch-all specifications and focusing on essential, learning-related data while providing an extension mechanism for customisation. The standard also provides naming conventions for institutions that implement the standard in its strict version (called "strictly conforming implementations") and extend it with some further detail ("conforming implementations"). This approach avoids describing all possible learner information, and includes only the minimum information necessary to satisfy functional requirements and to be maximally portable, with the ability to extend this information as needed.

4.2.3 Granularity and Flexibility

Another aim of this standard is to ensure different levels of granularity within information definitions. Following this advice, there are no coarse-grained information definitions, but rather a continuum, with many gradations and variations. For example, learner information can vary in "distance" from the trainee; local information, typically, is characterized by online availability, higher performance access,

and fewer security restrictions. There are also various possible degrees regarding privacy.

4.2.4 Requirements

PAPI Learner information must support:

- Cultural conventions (like measure units, currencies and other linguistic conventions).
- Institutional conventions. This mainly concerns learning institutions, each supporting its own conventions (like grading systems, or course denominations). An engineering goal of PAPI Learner is to "let the market solve the problem" of choosing/reducing the number of coding schemes to the "right" level.
- Simple application paradigms, in order to promote adoption. That is, it should require minimal effort to incorporate PAPI Learner codings, APIs, and protocols into existing real-world applications.
- Other engineering requirements include:
  1. Controlling access to information to the extent necessary.
  2. Maximizing performance of accessing data.
  3. Supporting varying data and information structures.
  4. Supporting varying coding and extension mechanisms.
  5. Supporting varying information partitioning schemes.
  6. Letting the market choose the best coding scheme(s).
  7. Supporting varying types of online, "sometimes", and offline connectivity.
  8. Supporting varying geographic (nomadic) access to information.

4.2.5 Data Exchange

Learners' data can be exchanged in three different ways:

- By means of the external specification, i.e., only PAPI Learner coding bindings are used while some other data communication method is mutually agreed upon by data exchange participants.
- Using the control transfer mechanism to facilitate data interchange, e.g., PAPI Learner API bindings.
- Employing data and control transfer mechanisms. e.g., PAPI Learner protocol bindings.

The conceptual model of data access is composed of:

Conceptual Model	**Description**
Data Object Model	A data object is at least one of: a data element, or an implementation-defined object.
Data Storage Model	Data, including data sets, may be stored in a data object, as referenced by an identifier.
Data Retrieval Model	Data, including data sets, may be retrieved from a data object, as referenced by an identifier.
Data Typing Model	Data objects that are data elements have a data type. Data types may prescribe certain value spaces (e.g., domains), representation, encoding, storage, layout, conversion to other types, methods, and operations. The data type of PAPI Learner data elements is defined by this Standard.
Data Attribute Model	A data attribute is an implementation-defined object associated with a data object. These attributes themselves may be accessed as data objects. Note: Attributes are also known as "properties".
Data Repository Access Model	PAPI Learner bindings define access, if any, to data repositories.

Conceptual Model	Description
Data Repository Security Model	The Security Model is defined and bounded by a security perimeter, with the following features defined during implementation: (1) the boundary of security perimeter(s); (2) nature, type, and acceptable level of risk of inbound security threats; (3) nature, type, and acceptable level of risk of outbound security threats; (4) security strength; (5) parameterisation, set-up, negotiation, and knockdown of security features, and (6) administration of the security perimeter integrity.
Data Persistence Model	The lifetime of data objects is implementation-defined.
Data Navigation Model	The techniques for navigating data structures are defined in PAPI Learner bindings.
Data Identification Model	The identification, labelling, namespace, and their associated techniques are implementation-defined.
Data Referencing Model	A data repository may create a reference to a data object for the purpose of subsequent de-reference. The naming conventions, lifetime, and scoping of a reference are implementation-defined.
Data De-referencing Model	A data repository may access a data object based upon supplying a reference, i.e., de-referencing a reference. The de-referencing methods are implementation-defined.
Data Indexing Model	The indexing methods for data repositories are implementation-defined. Note: The term "indexing" is used in the context of database systems, i.e., methods for organizing database records.
Data Searching Model	The searching methods for data repositories are implementation-defined.

4.2.6 Security

The following security features and concepts are part of the conceptual model definition:

- *Session-View-Based*. Security features are provided on a per-session, per-view basis. Each security session is initiated by an accessor (a user or agent that requests access). The accessor provides security credentials that authenticate the accessor, authorize the accessor, or both. A view represents a portion of PAPI Learner information and is similar to the notion of a database "view". Each view established represents a session, i.e., the "session" represents the duration of access and the "view" represents the scope of access.
- *Security Parameter Negotiation*. Data interchange participants negotiate security parameters prior to, during, and after each session. The security parameters are defined in the PAPI Learner bindings.
- *Security Extension*. Additional security features may be used that were not foreseen. The method of incorporating security extensions is defined in the PAPI Learner bindings.
- *Access Control*. Accessors may attempt read or write access to data elements, to create new data elements (separately or within aggregates), to destroy data elements (separately or within aggregates), and/or to change attributes of data elements. Other access methods, if any, are implementation-defined.
- *Identification*. The methods for identifying learners are implementation-defined. A related standard, IEEE 1484.13 ("Simple Human Identifiers"), defines the data type associated with a learner identifier.
- *Authentication*. The methods of authenticating users are outside the scope of the PAPI Learner Standard.

- *De-identification*. PAPI Learner prescribes that all information, except learner personal information, should be de-identified (that is, students should not be identified). The methods of de-identifying learners and their information are outside the scope of this Standard.
- *Authorization*. The methods of authorizing operations are implementation-defined.
- *Delegation*. The methods of delegating administration, authority, or credentials are implementation-defined.
- *Non-Repudiation*. The methods of non-repudiation are implementation-defined.
- *Repudiation*. The methods of repudiating data, users, or credentials are implementation-defined.
- *Privacy*. This Standard supports security frameworks and approaches that permit the implementation of a wide variety of privacy frameworks.
- *Confidentiality*. This Standard supports access controls and the partitioning of information types that permit the implementation of a wide variety of confidentiality frameworks.
- *Encryption*. PAPI Learner supports several security frameworks and techniques that permit the integration of various encryption models and technologies.
- *Data Integrity*. This Standard supports information assurance frameworks and approaches that permit the implementation of a wide variety of data integrity frameworks.
- *Validation of Certificates*. PAPI Learner does not require validation of student performance information, but supports the parameterisation of automated validation.
- *Digital Signature*. This Standard adopts third-party signing frameworks harmonized with ISO/IEC 15945.
- IEEE 1484.2.3, PAPI Learner Information Security Notes, contains information about applying security techniques and technologies to PAPI Learner implementations.

4.2.7 Data Representation

PAPI Learner supports many basic data types, like multilingual strings, arrays and so forth. They can be manipulated by a variety of operators (like create, destroy, move or search, etc.)

The following is a concrete example of data set ISO/IEC 11404–compliant for the data type "`PAPI_learner_bucket_type`" (user-defined enumerations of finite pairs name, value):

```
// ISO/IEC 11404 data set
(
 name = "parameter_1",
 value = "xyz",
),
```

An example of XML data instance for the same data type:

```
<!-- XML data instance ("..." is replaced by outer tags) -->
<...>
 <name> paramter_1</name>
 <value>xyz</value>
</...>
```

An example of Dotted Name-Value Pair (DNVP) data instance for the same data type:

```
#### DNVP data instance ("..." is replaced by outer context)
```

```
....name: parameter_1
....value: xyz
```

The DNVP notation is based on an RFC 822 style of messaging. Intuitively, it consists of parameter-value pairs, and is widely used in Internet-related communication protocols (web servers, e-mail systems, etc.).

4.3 IMS Learner Information Package (LIP)

Another major standardization effort, the Learner Information Package (LIP), comes from the IMS, a consortium of institutions including government agencies, software developers and vendors, and training and education representatives. Version 1.0 of the IMS Learner Information Package Specification was released to the public in March 2001. The IMS LIP has partly been derived from the IEEE PAPI Learner (versions 5.0 and 6.0).

The LIP specification provides a way of packaging learner information for exchange between disparate systems. It focuses on learner information, that is, the wide range of information that can be used by different systems to support the learner's activities. The semantics of the packages being exchanged may vary depending on the context; this is determined by the services participating in the exchange. Furthermore, learner information can be packaged from a variety of environments, not only human resources, student information and learning management systems.

An important aspect of the implementation of the XML-based specification to note is that nearly all LIP elements are optional. Depending on needs, data can be packaged to match the basic LIP segment structure or to match the structure of information on either side of the exchange. Either approach is acceptable.

LIP can be used for individual learner information packaging (for example, a student submitting his or her resume to an e-learning website) or for organizational exchange (both intra-organization, like data about employees, or extra-organization, like the certification of a student's achievements to a third-party institution).

Figure 3 below shows the IMS LIP data structure.

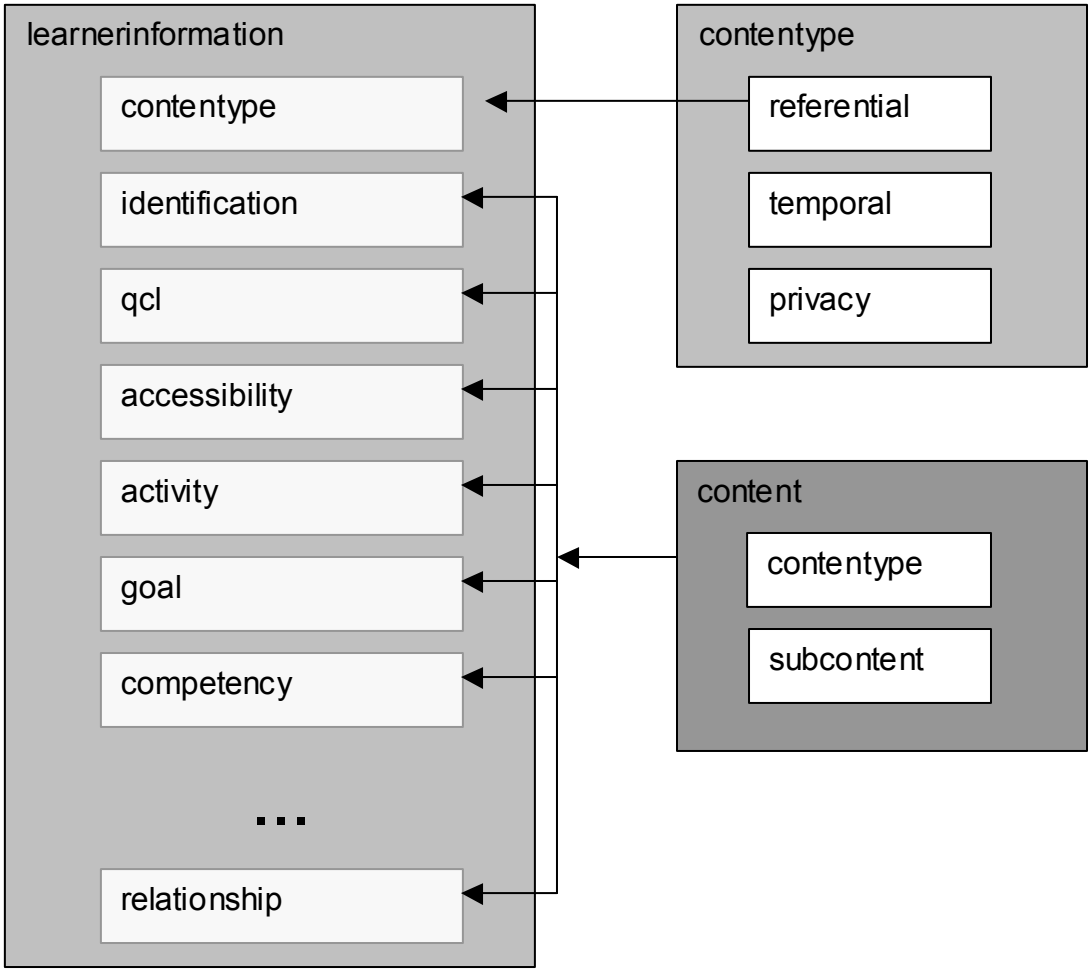

Figure 3. The LIP Data Model

The data structures that form the core of the IMS LIP specification are briefly outlined below.

4.3.1 Core Data Structures

LIP is structured around eleven core data structures, as follows:

1. **Accessibility** – Data regarding the accessibility of learner's information as defined through:
   - Language: the definition of a learner's language proficiencies.
   - Preference: the definition of a learner's cognitive, physical and technological preferences.
2. **Activity** – The activity the learner is engaging in, comprising:
   - Learning activity reference: an external reference mechanism to the learning materials.
   - Definition: the definition of the materials studied.
   - Product: the materials developed by the learners themselves.
   - Testimonial: statements attesting to the capabilities of the learner.
   - Evaluation: the results of the evaluations undertaken.
3. **Affiliation** – The learner's professional affiliations and associated roles.
   - **Competency** – The competencies of the learner.
   - **Goal** – The learner's goals and sub-goals.

4. **Identification** – The learner identification data. They comprise:
   - Formatted Name: the learner's name, formatted.
   - Name: the learner's name.
   - Address: the learner's addresses.
   - Contact info: electronic-based contact information about the learner.
   - Demographics: demographics information about the learner.
   - Agent: the representatives permitted to act on behalf of the learner.
5. **Interest** – Hobbies and recreational interests of the learner.
6. **Qcl** – A description of the qualifications, certifications and various licenses of a learner.
7. **Relationship** – the set of relationships that are to be defined between the learner and their identification, accessibility, qualifications, competencies, goals, activities, interests, transcripts, security keys and affiliations.
8. **Security key** – the security-related information for the given learner.
9. **Transcript** – the transcripts that summarize the performance of the learner.

A full, detailed list of all LIP data elements would be of little interest. What is important is that the standard has been designed to be extensible, in order to accommodate any possible learner data. Of course, the extensions obtained would be proprietary additions.

4.3.2 Relationship with the IEEE LTSC PAPI Specification

As mentioned earlier, the IMS LIP work incorporated the IEEE PAPI specification. Figure 4 describes such the relationship.

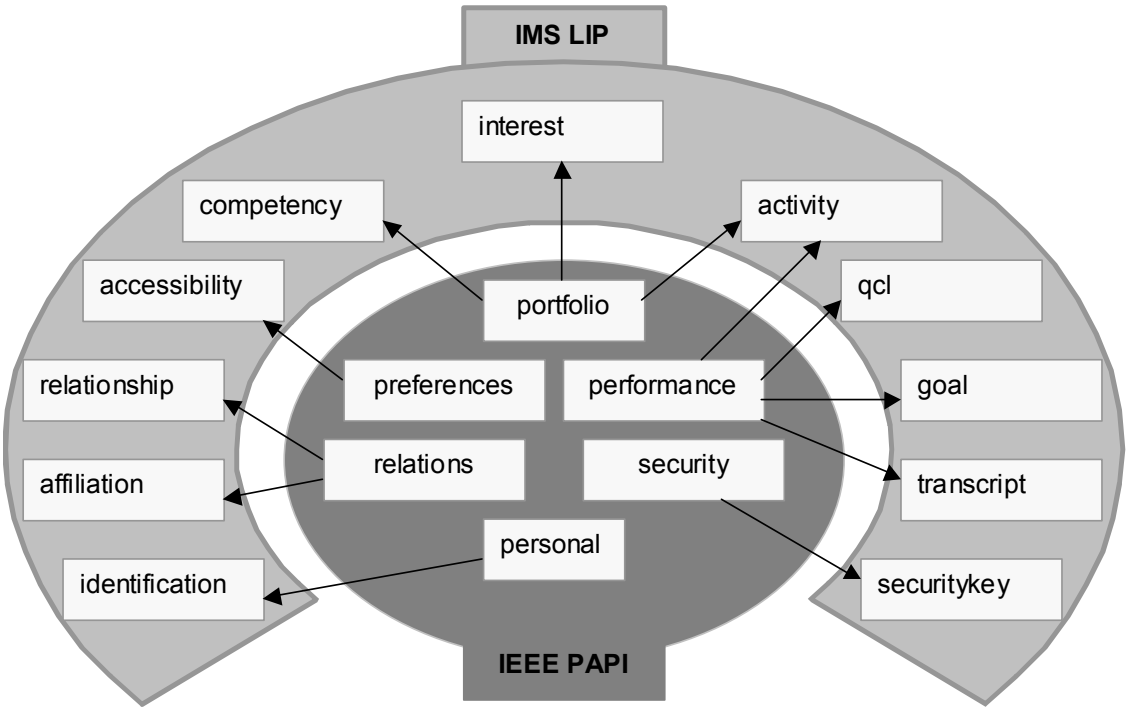

Figure 4. The Usage of IMS LIP to Support IEEE PAPI

An arrow in Figure 4 indicates the mapping between one data structure and another. Hence, data belonging to the IEEE PAPI `personal` group can be put in the `identification` IMS LIP data group when using the latter specifications.

4.3.3 Compatibility with the IMS Content Packaging Specification

The synergic use of IMS LIP and IMS CP is important in real-world e-learning systems, and is discussed here.

The IMS Content Packaging specification can be used for the packaging of a LIP XML instance for a single learner and for the aggregation of several instances of a single learner or for multiple learners. For example, in a case where the LIP-XML instances for two learners are to be packaged:

- The sets of information about learners have to be created, i.e. files 'Student1.xml' and 'Student2.xml'. In each case the learner information has some associated files: a meta-data file and other material files.
- The 'Photo.gif' files for the learners;
- Some proprietary data - the 'dataFile.dat' files.
- The 'Resume.doc' file, ensuring is does not cause a name clash.

The problem is ensuring that, when these learner information sets are packaged together, there are no clashes between the file names. The example is depicted in Figure 5 below.

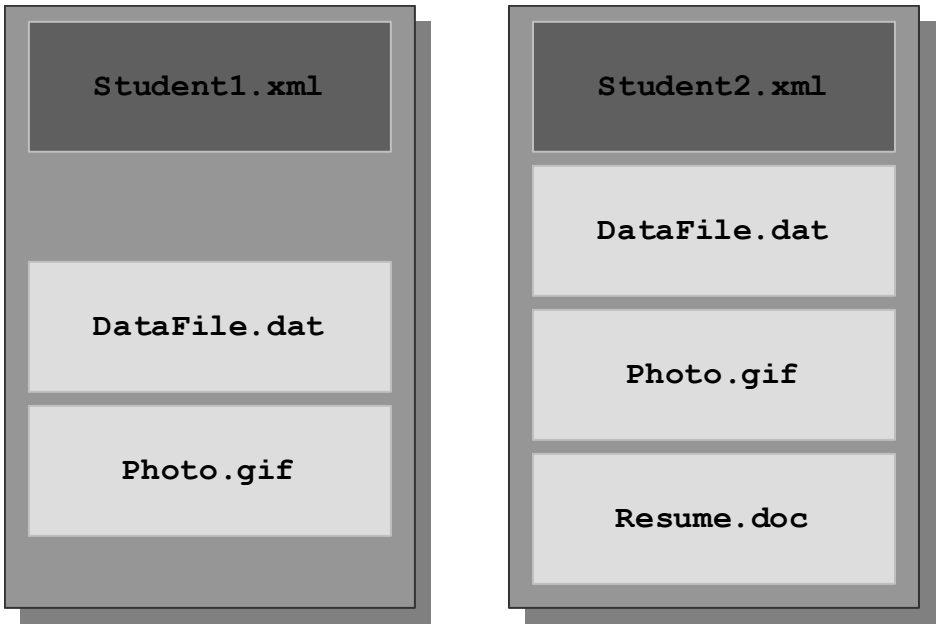

Figure 5. Schematic representation of the files to be packaged.

The IMS Content Packaging requires that all of the packaged files are uniquely named, i.e. an explicit file directory structure has to be used to ensure that file clashes do not occur when creating the package. The example below shows the XML used to refer to the 'Photo.gif' file references in the XML instances for Student1 and Student2.

The partial XML instance of Student1:

```
<representation>
    <media mediamode="Image" mimetype="image/gif" encoding="uri">
Photo.gif</media>
</representation>
```

The partial XML instance of Student2:

```
<representation>
    <media mediamode="Image" mimetype="image/gif" encoding="uri">
Photo.gif</media>
</representation>
```

The preferred packaging approach is as follows:

- In a Content Package create a sub-directory for each student with a unique name within the package;
- In that directory place the LIP instance and the associated portfolio files;
- If the portfolio files reference each other using relative sub-directories, then these should be maintained within the allocated sub-directory;
- References to portfolio files should be relative to their shared allocated sub-directory, i.e. top-level references will only need to be preceded by a slash but not the directory name and existing relative sub-directory names are also used as defined;
- A resource element is used to specify the file name of the LIP instance and all the associated portfolio files;
- The `xml:base` attribute of the LIP resource element is used to specify the relative address of the allocated sub-directory.

Finally, note that there is no metadata provision for IMS LIP. The exchange of IMS LIP will be achieved through the packaging of the XML instance and all of its associated files using the IMS Content Packaging specification mentioned here briefly. Because IMS CP includes the use of metadata, adding it to the IMS LIP would have been redundant. The usage of the metadata within the IMS Content Packaging specification is thought to be sufficient, hence no metadata is included within the IMS LIP specification itself.

4.3.4 Some Examples

After having introduced the theory behind the IMS LIP specifications, we conclude with some concrete examples of its usage.

4.3.4.1 An Accessibility Example

We saw that the accessibility group gathers data regarding the accessibility of learner's information such as language (the definition of a learner's language proficiencies) and preference (the definition of a learner's cognitive, physical and technological preferences). The following example concerns a student that speaks excellent Italian, has a good reading ability in this language, but has poor writing skills.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE learnerinformation SYSTEM "ims_lipv1p0.dtd">
<learnerinformation>
      <comment>LIP Accessibility information.</comment>
      <contentype>
            <referential>
                  <sourcedid>
                        <source>Learner_Core_Data</source>
                        <id>example_0001</id>
                  </sourcedid>
            </referential>
      </contentype>
      <accessibility>
            <contentype>
                  <referential>
                        <indexid>accessibility_0001</indexid>
                  </referential>
            </contentype>
            <language>
                  <typename>
                        <tysource sourcetype="imsdefault"/>
                        <tyvalue>Italian</tyvalue>
                  </typename>
                  <contentype>
                        <referential>
                              <indexid>language_01</indexid>
                        </referential>
                  </contentype>
                  <proficiency profmode="OralSpeak">Excellent</proficiency>
                  <proficiency profmode="Read">Good</proficiency>
                  <proficiency profmode="Write">Poor</proficiency>
            </language>
      </accessibility>
</learnerinformation>
```

4.3.4.2 A Certification Example

A common issue when modelling concrete students concerns the provision of data about the certifications they obtained. We saw that LIP provides the QCL element for this purpose. The following example describes a student who possesses, among other certificates, an "A" scuba diving license.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE learnerinformation SYSTEM "ims_lipv1p0.dtd">
<learnerinformation>
      <comment>Certification Example using QCL.</comment>
      <contentype>
            <referential>
                  <sourcedid>
                        <source>Learner_Data</source>
                        <id>cert_0001</id>
                  </sourcedid>
            </referential>
      </contentype>
      <qcl>
            <typename>
                  <tysource sourcetype="imsdefault"/>
                  <tyvalue>Certification</tyvalue>
            </typename>
            <contentype>
                  <referential>
                        <indexid>certif_0001</indexid>
                  </referential>
            </contentype>
            <title>Scuba Diving A</title>
            <organization>
                  <typename>
                        <tysource sourcetype="imsdefault"/>
                        <tyvalue>Training</tyvalue>
                  </typename>
                  <description>
                        <short>Scuba Diving International Institute</short>
                  </description>
            </organization>
            <registrationno>SSN_987654</registrationno>
            <date>
                  <typename>
                        <tysource sourcetype="imsdefault"/>
                        <tyvalue>Award</tyvalue>
                  </typename>
                  <datetime>2001:011</datetime>
            </date>
      </qcl>
</learnerinformation>
```

The student (referenced to by the "`cert_001`" symbolic id) in the above example received this certificate from the "Scuba Diving International Institute" in 2001; the related certificate has a registration number "SSN_987654".

4.3.4.3 An Example of Data Security.

The `securitykey` element describes security-related information for the given learner.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE learnerinformation SYSTEM "ims_lipv1p0.dtd">
<learnerinformation>
      <comment>An example of LIP Securitykey information.</comment>
      <contentype>
            <referential>
                  <sourcedid>
                        <source>Sec_Example</source>
                        <id>sec_id_0001</id>
                  </sourcedid>
            </referential>
      </contentype>
      <securitykey>
            <typename>
                  <tysource sourcetype="imsdefault"/>
                  <tyvalue>Password</tyvalue>
            </typename>
            <contentype>
                  <referential>
                        <indexid>id_seckey_1</indexid>
                  </referential>
            </contentype>
            <keyfields>
                  <fieldlabel>
                        <typename>
                              <tyvalue>Password</tyvalue>
                        </typename>
                  </fieldlabel>
                  <fielddata>myPazzWd99</fielddata>
            </keyfields>
      </securitykey>
</learnerinformation>
```

The student (referenced by the "sec_id_0001" symbolic ID) in the example above has a password code "myPazzWd99".

4.4 Saba Profile Format

Profile Format (http://www.saba.com/standards/ulf/Specification/specPROF.htm) is an XML-based representation for describing learner profile information. Learner profiles comprise a variety of data about learners, including personal and job information, learning history, goals and plans, and held competencies and certifications. Profile Format captures this information in an XML-based format using RDF to define metadata for describing learners. Profile Format incorporates several existing metadata standards, including the Dublin Core and vCard, which ensures compatibility with existing person/profile descriptions.

By employing Profile Format to describe the learners in a system, learning providers can extend their learning management architecture to support all of the following:

- Searches of critical learner metadata such as name, title, role, learning results, and held competencies and certifications
- Tracking the learning history of individual learners
- Assignment of competencies (with proficiency levels) and certifications to learners
- Assignment of learning goals to learners and tracking of progress towards fulfilment of those goals

- Creation of distributed profiles, where portions of a learner's profile are provided by different sources
- Compatibility with standard web search engines

Profile Format is based on open standards and is designed to reflect the following principles:

- Compatibility with emerging industry standards for learning profiles, including ongoing work in IMS and IEEE.
- Extensibility to easily accommodate future growth and change.

Profile Format employs an XML standard known as RDF (Resource Description Framework), the standard for defining metadata to describe web-based resources. The use of RDF makes it possible to define a set of unique RDF properties and merge these properties with properties defined in existing standards, such as vCard and Dublin Core. RDF also provides a unified mechanism for manipulating and querying this merged metadata. Furthermore, the use of RDF allows Profile Format to support distributed profiles, where portions of a learner's profile are provided by different sources.

4.4.1 Description Element

A Profile Format document is an RDF document that contains one or more **Description** elements, where each element describes a learner in the system. Each **Description** element contains a unique identifier and a set of property/value pairs that fully describe the learner. These properties can draw from any of the Profile Format RDF schemas.

Each **Description** element has an attribute that unambiguously identifies the learner being described. This attribute can be either of the following:

- id
- about

The **Description** element can also include the **xml:lang** attribute for specifying the language in which the metadata description is authored. The **xml:lang** attribute contains the ISO 639 /RFC 1766 language code with an optional geographic identifier, such as **en** for English, or **fr** for French.

Using the "id" Attribute

The **id** attribute specifies a unique ID for the learner, for example:

```
<rdf:Description id=”sally_brown” xml:lang=”en-US”>
...
</rdf:Description>
```

Using the "about" Attribute

The **about** attribute specifies the URL of a resource, for example:

```
<rdf:Description about=”http://www.saba.com/people/sally_brown” xml:lang=”en-US”>
...
</rdf:Description>
```

The **about** attribute is useful when the metadata description applies to learner information defined in some other location, such as a home page.

4.4.2 Profile Format Properties

Profile Format subdivides learner information into the following categories:

- Personal information. This includes information such as name, address, title, role(s), and organisation membership. All personal information is represented using RDF mappings of the vCard specification.

- Learning information. Profile Format defines a set of RDF properties that capture information on a learner's current learning (current enrolments) and learning history (transcript). The **learning** property specifies a URL to a learning resource described in a Catalog Format document. The specified resource is a held offering in the learner's transcript. In its simplest form, the **learning** property contains only the URL of the held learning offering, for example:

  <profile:learning rdf:resource="http://www.saba.com/learning/catalog.rdf#JAVA101"/>

  The **learning** property can also be a structured property, with substructure properties that qualify the status of the learning offering and the conditions under which it was attained. For qualified instances of the **learning** property, the URL of the learning resource is captured using the **rdf:value** property.

- Goal information. Profile Format defines a set of RDF properties that capture information about a learner's goals. Goals can encompass both business and professional objectives for a learner and include the following additional information:

  - planned actions for achieving the goal
  - learning interventions
  - accomplishments

  The **goal** property defines a learner's goal. In its simplest form, the **goal** property contains the name of an in-progress goal and has an optional **id** attribute, for example:

  <profile:goal rdf:id="JAVA1">Become a Java expert</profile:goal>

  Alternatively, the goal can be represented as a reference to a particular competency level, certification, or learning intervention. In these cases the **resource** attribute is used to reference the relevant URL:

  <profile:goal rdf:resource="http://www.saba.com/competencies/programming#Java.Expert"/>
  <profile:goal
  rdf:resource="http://www.saba.com/certifications/certifications.xml#MS
  CD"/>
  <profile:goal rdf:id="VJ++"
  rdf:resource= "http://www.saba.com/courses/ms.rdf#VJ60">
  Learn Visual J++
  </profile:goal>

The **goal** property can also be a structured property, with substructure properties that provide details about the goal and its status. For qualified instances of the **goal** property, the URL of a qualified goal is captured using the **rdf:value** property.

- **Observation** information. Profile Format defines a set of RDF properties that capture information reflecting a learner's progress towards specific goals. These observations track specific, measurable milestones on the path towards achieving a goal.

- **Competency** information. Includes information on held competencies. This category contains a pointer to a competency defined in a Competency Format document, with optional properties describing how the competency was attained.

- **Certification** information. This category contains a pointer to a certification track defined in a Certification Format document, with optional properties describing how the certification was attained.

- **Preference** information. Profile Format defines a set of RDF properties that capture information reflecting the learning preferences of a learner. Includes information on learner preferences, such as home language and country.

- **Profile** Information. Includes information about the profile itself, such as the date it was generated and the language it is in. All profile information is represented using the RDF mappings of Dublin Core.

4.4.3 Example

The following Profile Format document represents a complete profile for a learner ("Sally Brown") and incorporates properties from several of the namespaces supported by Profile Format.

```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:vCard = "http://imc.org/vCard/3.0#"
      xmlns:profile="http://www.saba.com/RDF/profile/1.0#"
      xmlns:cert="http://www.saba.com/RDF/certification/1.0#">

<Description about="http://www.saba.com/people/sally_brown">
<rdf:type rdf:resource="http://www.saba.com/RDF/profile/1.0#learner"/>

<!-- personal information -->
      <vCard:FN>Sally Brown</vCard:FN>
      <vCard:ROLE> Programmer </vCard:ROLE>
      <vCard:ORG rdf:parseType="Resource">
            <vCard:Orgname>Saba Software</vCard:Orgname>
            <vCard:Orgunit>
                  <Seq>
                        <li>Research and Development</li>
                        <li>Platform Group</li>
                  </Seq>
            </vCard:Orgunit>
      </vCard:ORG>

<!-- learning information -->
      <profile:learning>
                              <Bag>

            <!-- current learning -->
                        <li parseType="Resource">
                              <value
                              resource="http://www.saba.com/learning/catalog.rdf#Banking101.2"/>
                              <profile:status>Enrolled</profile:status>
                        </li>

            <!-- learning history -->
                        <li resource="http://www.saba.com/learning/catalog.rdf#MathIntro.1"/>
                        <li resource="http://www.saba.com/learning/catalog.rdf#Philosophy.5"/>
                  </Bag>
            </profile:learning>

      <!-- goal information -->
                        <profile:goal>
        <Bag>
              <li>Become a Java expert</li>
              <li parseType="Resource">
```

```
                        <value>Reduce bug count by 50%</value>
                        <profile:status>In Progress</profile:status>
                    </li>
                    <li parseType="Resource">
                        <value>Learn CVS</value>
                        <profile:status>Completed</profile:status>
                    </li>
                </Bag>
            </profile:goal>

        <!-- held certifications -->
            <cert:certification>
                <Bag>
                    <li resource="http://www.saba.com/certifications/certifications.xml#MSCD"/>
                    <li resource="http://www.saba.com/certifications/certifications.xml#NOVL"/>
                </Bag>
            </cert:certification>

        <!-- preferences -->
            <profile:language>en</profile:language>
            <profile:country>US</profile:country>

    <!-- profile information -->
        <profile:publisher>http://www.saba.com</profile:publisher>
        <profile:date>2000-04-25</profile:date>

    </Description>
    </RDF>
```

4.5 Some E-Learning Systems

Having introduced the major student modelling standards, we will briefly examine how they have been implemented on a practical level.

4.5.1 Blackboard

Only academic works and research systems for computer-based training have been illustrated so far. This overview would not be complete, however, without mentioning the leading commercial products. They currently account for the largest market slice (i.e. the student population that is actually being helped by intelligent tutoring systems) by far.

Blackboard [Blackboard01] is a complete suite of products for e-learning developed by Blackboard Inc. It provides a wide range of features related to e-learning, including:

- Course Management. The system allows content creation, management, sequencing and delivery, both on the Web and on a stand-alone PC, for all the most popular platforms.
- Online Community and organizational management. This allows customers to build specialized portals while supporting services like web-based email, community-building and overall organizational management.
- System Management. Blackboard provides many facilities for developers and system administrators for easy and seamless integration with the pre-existing software environment, plus a host of further technical features.

Blackboard is one of the most widely used e-learning solutions, claiming more than 1,800 clients

worldwide. Furthermore, it supports interoperability with other current competing products; the company is actively involved in standards specification committees.

The adopted student model is a direct implementation of the IMS specifications that we have already seen. Technologically, its implementation is based on state-of-the-art technology. Its architecture is a three-tiered one, relying on back-end databases, a Java-based middle-tier and a variety of front-ends for end-user interaction.

4.5.2 ILIAS

We present here one of the first open-source initiatives aimed at innovative e-learning solutions. ILIAS has been developed in the VIRTUS project of the Faculty of Economics, Business Administration and Social Sciences at the University of Cologne. The ILIAS software is available as open source software under the terms of the GNU-GPL. The authors are working to make the system adhere to all major standards.

Having been developed and used in the University environment, this system differs from commercial products, for example, with regard to the roles involved. ILIAS recognises the following roles:

- Learner
- Author (of learning-material)
- Administrator
- Researcher
- Guest

Along these lines the system offers all management functions found in usual e-learning suites. What is interesting to examine, though, is the learning approach that inspired the system. The authors have modelled the system on the "moderate constructivism" learning approach.

One way to achieve a student's active participation (thus following the constructivist approach) is to encourage the reader to annotate the text and to share these annotations with virtual learning communities. For this purpose ILIAS offers a context related annotation function that permits adding summaries, comments and annotations to text elements, graphics or images. These annotations are saved on a personal level (so they are part of the ILIAS student model) and can be retrieved at any time within the original context. At the end of a course learners can print these annotations out in a structured form, or exchange them with members of their learning group. Furthermore, the same is possible for documents outside the course scope. By way of a personal bookmark folder learners are encouraged to mark and organise related resources available on the Internet. Students are thus prompted to conduct further, autonomous research, relating individual lessons to other contexts. This way, knowledge is subjected to reinterpretation and reconstruction, by means of the relatively cheap registration of personal annotations in the student model.

Also following the constructivist approach, learners can structure their work independently, drawing upon the system via the Internet which provides access to the respective hypermedia courses consisting of the learning material and individual annotations.

Constructivist concepts emphasise the importance of so-called knowledge building discourses as methods of knowledge acquisition. Accordingly, context-oriented communication is of fundamental importance; that is why ILIAS offers its communication features within and related to specific contexts. The context sensitivity allows for immediate interaction between individual learners who are facing a specific set of problems at a certain time. The ILIAS system can monitor and initialise communication about problems relating precisely to the actual and current (learning) context.

4.6 Other Initiatives

This section covers other standardization initiatives that use student modelling, however marginally.

4.6.1 The European Initiative for E-learning

The CEN/ISSS Workshop on Learning Technologies (WSLT) has commissioned a survey of current Educational Modelling Languages (EMLs). The Study is lead by The Open University of the Netherlands, with the participation of The British Open University and UNED (Universidad Nacional

de Educacion a Distancia, Madrid, Spain).

EMLs are being developed and used around the world. They tend to be based on XML and are used to create highly-structured course material. An EML-based course might offer features such as: re-useable course material, personalized interaction for individual students, media independence, etc.

WSLT is working on a reparatory agreement that may ultimately lead to an official European standard. Through standardization, it would be possible, for example, for course material to be re-used between standard-compliant platforms from different vendors. Unfortunately, this initiative is still in its early stages and it will possibly take several years for such a standard to be issued.

4.6.2 Schools Interoperability Framework (SIF)

The Schools Interoperability Framework (SIF) is an industry initiative to develop a technical blueprint for K-12 software that will enable diverse applications to interact and share data now and in the future. It addresses the variety of data exchange needs in school districts. As the first component, SIF 1.0 focuses primarily on student and transportation data. Future versions of the SIF specification will expand upon the type of data that can be exchanged.

SIF has two deliverables: the SIF Message Specification (for defining the messages that each application can exchange with others) and the Implementation Specification (that defines the software implementation guidelines for SIF). This latter specification does not make any assumption of what hardware and software products need to be used to develop SIF-compliant applications. Instead, it only defines the requirements of architecture, communication, software components, and interfaces between them.

The goal of SIF is to ensure that all SIF-compliant applications can achieve interoperability, regardless of how they are implemented. SIF is truly an open industrial initiative. It is focused on supporting interoperability between (North-American) schools-based educational administration systems whereas initiatives like LIP are focused on learner educational information.

4.6.3 ANSI TS 130 Student Educational Record

The ANSI TS130 contains the format and establishes the data contents of a Student Educational Record (Transcript) Transaction Set for use within the context of Electronic Data Interchange (EDI) environment. The student transcript is used by schools and school districts, and by post-secondary educational institutions to transmit current and historical records of educational accomplishment and other significant information about students enrolled at the sending schools and institutions. The transcript may be sent to other educational institutions, to other agencies, or to prospective or current employers. The student transcript contains personal history and identifying information about the student, their current academic status, dates of attendance, courses completed with grades earned, degrees and diplomas awarded, health information (Pre-Kindergarten through Grade 12 only), and testing information.

4.6.4 Internet vCard Specification

The vCard specification allows the open exchange of Personal Data Interchange (PDI) information typically found on traditional paper business cards. The specification defines a format for an electronic business card, or vCard. Such a specification is suitable as an interchange format between applications or systems; its format is defined independently of the particular method used to transport it (it supports a file system, point-to-point public switched telephone networks, wired-network transport, or some form of unwired transport). The vCard can be used to forward personal data in an electronic mail message, or for automating the filling out of web-based forms in HTML pages, etc.

4.6.5 Internet2 eduPerson

The eduPerson specification, created by EDUCASE and Internet2, aims at services that provide seamless access to network-accessible information, regardless of where or how the original information is stored. It achieve this by providing a set of standard higher-education attributes for an enterprise directory, which facilitate inter-institutional access to applications and resources across the higher education community. It is a technologically–driven specification, in that EDUCAUSE/Internet2 eduPerson task force has the mission of defining a Lightweight Directory Access Protocol (LDAP)

object class that includes widely-used person attributes in higher education.

4.6.6 HR-XML Consortium Specifications

The HR-XML Consortium is an independent, non-profit association dedicated to the development and promotion of a standard suite of XML specifications for permitting e-commerce and the automation of human resources-related data exchanges. The mission of the HR-XML Consortium is to develop and publish open data exchange standards based on XML.

Some of the Consortium's initial targets for standardization activities include recruiting, staffing, compensation and benefits. The Consortium's Recruiting and Staffing workgroup is working on a first version of Staffing Exchange Protocol (SEP), an XML-based messaging framework that supports dynamic, real-time staffing transactions over the Web. Such a protocol will allow for operations like job opportunities postings to job boards, related updating, recall and searches, etc.

4.6.7 Universal Language Framework (ULF)

ULF is a proprietary standard from Saba inc. that complies with IMS standards as regards student modelling. ULF utilizes many of the industry standards for exchanging learning data in a Web environment (including ADL, IMS, LRN, IEEE LTSC, Dublin Core, and vCard) bringing together the key elements of these standards into a comprehensive and fully integrated solution.