A framework for learner modelling

Pierre Dillenbourg, John Self

Interactive Learning Environments, 1992, 2 (2), 111-137.

A FRAMEWORK FOR LEARNER MODELLING

Pierre DILLENBOURG[1]
John SELF[2]

[1]Faculté de Psychologie et des Sciences de l'Education, Université de Genève, 1211 Genève 4, Switzerland
[2]Department of Computing, Lancaster University, Lancaster LA1 4YR, England

ABSTRACT

This paper presents a comprehensive conceptual framework and notation for learner modelling in intelligent tutoring systems. The framework is based upon the computational distinction between behaviour, behavioural knowledge, and conceptual knowledge (in a 'vertical' dimension) and between the system, the learner, and the system's representation of the learner (in a 'horizontal' dimension). All existing techniques for learner modelling are placed within this framework. Methods for establishing the search space for learner models and for carrying out the search process are reviewed. The framework makes clear where particular learner modelling techniques are focussed and shows that they are often complementary since they address different parts of the framework.

A FRAMEWORK FOR LEARNER MODELLING

Pierre Dillenbourg and John Self

1. THE FRAMEWORK

Learner models are important within computer-based systems intended to promote learning because they provide the means to support individually-adapted instruction. The task of 'learner modelling' or 'cognitive diagnosis' (that is, the task of building a learner model) is defined later, but, in short, it is the process of inferring the learner's knowledge by analysing his or her behaviour.

Although a number of learner modelling techniques have been developed, learner modelling remains a serious problem for the implementation of computer-based educational systems. Simply enumerating these techniques does not really help us: we need a framework describing precisely the role of these techniques in order to discuss their adequacy. As we will see, it turns out that many of the techniques are not alternatives but address different aspects of the learner modelling problem.

The aim of this report is to provide a comprehensive conceptual framework for learner modelling. The framework we propose includes a formalism, a terminology and a graphical representation of the learner modelling process.

The purposes of this framework are :

1. to define a consistent terminology for use in research on learner modelling,
2. to describe several approaches to learner modelling and to make precise their particular focus,
3. to describe especially the contributions of machine learning techniques,
4. to emphasize the importance of conceptual aspects in learner modelling,
5. to identify (implicit) underlying assumptions.

1.1 The vertical dimension

Vertically (see Figure 1), the framework discriminates three related entities. The relationship between these entities is that they all concern the same *problem domain* . If we adapt the definition proposed by Van de Velde (1988), a problem domain is a triple $\Omega = (P, B, Solution)$ where:

P is the set of problems in Ω

B is the set of possible behaviours in Ω

Solution is a relation between P and B, i.e. Solution is a subset of $P \times B$.

A *behaviour* (B) is defined as a sequence of actions performed by an agent (A) as a potential solution for some problem (P) belonging to P. A behaviour will be described by the following syntax: behaviour(agent,problem) = $B_A P$. The agent may be the learner or the computer system. B, the set of possible behaviours, is generally very large, since it includes any correct, incorrect, or even inconsistent behaviour.

The *behavioural knowledge* (bk) of the domain corresponds to Van de Velde's concept of *problem solver*. A problem solver contains an inference structure and a logical theory. The inference structure is a set of *inferential primitives* and *inference relations* to be used to infer some behaviour B (from B) for a given problem P (from P).

The *conceptual knowledge* (ck) contains the definition of the concepts underlying the behavioural knowledge. The discrimination between the behavioural and conceptual knowledge corresponds to the "shallow" vs "deep" knowledge issue raised in expert system design (as will be discussed below). The deep knowledge consists of domain theories and problem solving knowledge; the shallow knowledge is a procedural description of the problem solutions.

As shown in Figure 1, the conceptual knowledge may exceed the problem domain Ω, i.e. it may enclose concepts which are usable for problems not present in P. In the following example, ck's concepts such as "disease" or "symptoms" are not restricted to the bk's context of heart disease diagnosis. For example :

Ω	the problem of heart disease diagnosis
$B_D P$	the diagnosis produced by a doctor for a particular patient
bk	set of Mycin-like rules for diagnosis and the inference engine
ck	representation of the heart, concepts for diseases, etc.

< about here: Figure 1. The vertical dimension of the framework >

Globally, the vertical relation is one of **consistency**. The relation between the behavioural knowledge and the behaviour has a level of consistency dependent on the logical theory inscribed in the behavioural knowledge. This relationship may be viewed top-down as a "*running* " process: the behaviour is the result of running the behavioural knowledge on a particular problem of P. The relationship between the conceptual and behavioural knowledge may be viewed top-down as some *knowledge compilation* process.

The consistency between two entities will be denoted in our framework formalism by the symbol ¢ (0<¢<1), where ¢=1 denotes complete consistency (thus ¢ denotes a measure of the agreement between two entities, not strict logical consistency, which would of course be just true or false):

$\phi(bk, B_A P)$ indicates the consistency between the behavioural knowledge and A's behaviour for problem P.

$\phi(ck, bk)$ indicates the consistency between the conceptual and behavioural knowledge sets.

$\phi(B_A P, ck)$ indicates the consistency between A's behaviour for problem P and his conceptual knowledge.

As we will see, most work on learner modelling postulates high values for the various ϕ. In general, the quality of the diagnosis built will be a function of the values of the ϕ. By definition, a behaviour results directly from bk and indirectly from ck, and hence we can state that normally $\phi(ck, B_A P) < \phi(bk, B_A P)$.

When we want to refer indifferently to bk or ck, we will use *k.

1.2 The horizontal dimension

By contrast to the vertical dimension, the horizontal one (see Figure 2) emphasizes **discrepancies** between the same entities possessed by different agents. The learner modelling context involves two agents, namely the learner and the system. The object of learner modelling is to build the best representation of the learner (we will indicate below what might be meant by 'best' in this context). Our model emphasizes the fact that this is only a computerised representation of the learner's knowledge, an approximation, and that the system has no direct access to his or her knowledge.

This horizontal dimension clearly indicates that our framework adopts a differential perspective. The term "*differential modelling*" is often used when the learner's behaviour and knowledge is represented with respect to the system's behaviour and knowledge. In the formalism used in this framework, we use the expression $R_A x$ to denote the representation that an agent A has of some object x.

So the model horizontally discriminates three entities :

S the system,

L the learner,

$R_S L$ the system's representation of the learner.

< about here: Figure 2. The horizontal dimension of the model >

A discrepancy will be represented by Δ. Three kinds of discrepancy may appear between these entities :

$\Delta(L,S)$ the difference between the learner and the system. (This is often considered to describe learner errors, since the system is usually supposed to be "correct")

$\Delta(S,R_SL)$ the difference between the system and the system's representation of the learner. (This may be regarded as the representation of $\Delta(L,S)$, built by the system, or in other words $\Delta(S,R_SL) = R_S\Delta(L,S)$.)

$\Delta(L,R_SL)$ the difference between the learner and the system's representation of the learner. (This represents the error in learner modelling, i.e. the difference between the actual learner and the diagnosis built by the system.)

There is a clear difference between 'discrepancies' and 'consistencies': discrepancies indicate differences between two similar entities (e.g. two behaviours), while consistencies emphasize the logical link between very different things (e.g. an agent's knowledge and his behaviour).

1.3 The model

The two described dimensions are crossed for forming the model : each entity of the vertical dimension exists for each entity of the horizontal dimension (see Figure 3). The model contains the following components:

R_Sck the system's (representation of) conceptual knowledge.

R_Sbk the system's (representation of) behavioural knowledge.

B_SP the system's behaviour on problem P.

R_Lck the learner's (representation of) conceptual knowledge.

R_Lbk the learner's (representation of) behavioural knowledge.

B_LP the learner's behaviour on problem P.

R_SR_Lck the system's representation of the learner's (representation of) conceptual knowledge.

R_SR_Lbk the system's representation of the learner's (representation of) behavioural knowledge.

R_SB_LP the system's representation of the learner's behaviour on problem P.

< about here: Figure 3. The model >

This set might be extended with representations such as :

R_LR_Lck the learner's representation of his own representation of the conceptual knowledge (metacognition).

R_LR_Sck the learner's representation of the system's conceptual knowledge (a crucial issue in interface design).

...

The word "system" might be replaced by "ITS", or more precisely by one of the ITS components. The standard ITS components are the expert (or domain model), the tutor (or pedagogical model), the diagnoser + learner model and the interface. When we are speaking about the knowledge required for solving a class of problems (i.e. $R_S bk$) or about the system's behaviour (i.e. $B_S P$), the "s" refers to the expert component. Similarly, the "s" in $R_S R_L bk$ is the diagnoser component. Such a discrimination among the system's components is however not pertinent in our formalisms because the diagnostic component highly depends on the expert knowledge for building the diagnosis.

The main *input* to the learner modelling process is $R_S B_L P$. The system's behaviour $B_S P$ and its representations of knowledge $R_S bk$ and $R_S ck$ constitute other inputs. If the learner model is not built *ex nihilo* but is incrementally adapted from a previous model, then $R_S R_L bk$ and $R_S R_L ck$ are also to be considered as items of input.

A major advance of ITS over traditional Computer Assisted Instruction is to take into account what the learner has and has not understood. This means that the main *output* of the learner modelling process is a description of $R_L bk$ and/or $R_L ck$. Globally, traditional CAI may be located at the first level of the model (behaviour), most ITSs reach the second level (behavioural knowledge) and some of them reach the third level (conceptual knowledge). The flavour of this model is to put the emphasis on the conceptual level.

To indicate that the representations of learner knowledge are diagnoses built from some behaviour, we write

$$R_S R_L{*}k = f(B_L P)$$

where f is some diagnostic function.

The *learner model* is, according to this model, $R_S R_L bk$ and/or $R_S R_L ck$, i.e. the output described above. The word "model" may have various significations, from a simple enumeration of characteristics or numeric parameters (as in traditional CAI) to an integrated set of knowledge, close to the idea of mental model proposed by cognitive scientists.

The *learner modelling process* (or diagnosis process) is the process of inferring a learner model (or diagnosis) from a learner's behaviour. A possible goal of the learner modelling process is to obtain the best image of $R_L ck$ and $R_L bk$, i.e. to minimize $\Delta(R_L ck, R_S R_L ck)$ and $\Delta(R_L bk, R_S R_L bk)$.

The system needs to be informed about the quality of its output, i.e. to receive some appraisal of the value of $\Delta(R_L{*}k, R_S R_L{*}k)$. We call this information *diagnosis feedback*. It is clearly different from the didactic feedback used in tutorial interactions: diagnosis feedback confirms to the system that its representation of the learner fits with the learner's representation, while didactic feedback, on the other hand, tells the learner how

his representation fits with the system's representation. We will later on describe several ways for obtaining this feedback :

- *behavioural prediction* : the learner's behaviours are used to confirm the diagnosis or to choose among current hypotheses,

- *explicit interaction* : the learner is asked to confirm the system's diagnosis,

- *behavioural simulation* : $R_S R_L bk$ is run to produce a predicted $B_L P'$ which is compared to $B_L P$ - if they match the diagnosis is confirmed (under the postulate that the running process is isomorphic to human reasoning).

- *didactic prediction* : if the learner model is valid, it can be used to select a didactic action, i.e. to predict the efficiency of this action or to anticipate the learner's knowledge after the action. If the predicted changes happen, the diagnosis is confirmed (this raises two problems which are described in section 4.4.2).

At this stage, we are of course glossing over a great many details and subtleties - most of these issues will, we hope, be addressed.

1.4 The model's context

Here we list some implicit assumptions which are generally present in learner modelling work :

Assumption 1 : Learner modelling is related to some idea of what tutoring is: ITSs need diagnostics for taking decisions, which means that it is the system which takes at least some of the didactic decisions, and not the learner. (A possible alternative justification for learner modelling is that it enables the ITS to show the learner his own knowledge, misconceptions, etc.)

Assumption 2 : The learner model is seldom exhaustively described (which is theoretically impossible given the infinite amount of common sense knowledge involved). More often, it is by default assumed to be closely related to the system's (conceptual or behavioural) model. This means that only the discrepancies between these two sets of knowledge need to be represented (hence the "diagnosis" term, defined in section 2.2.2). This is the basic assumption of any differential approach, but it is also a major point of criticism of work in learner modelling since many systems represent 'expert knowledge', which is known to be very different to that of learners.

Assumption 3 : Learners are often viewed as correctly applying an incorrect algorithm (i.e. in our formalism, $R_L bk$ is wrong but $¢(B_L P, R_L bk)$ is high), while instead most work in education considers the learners as incorrectly applying a correct algorithm.

Assumption 4 : Differential modelling proceeds as if there were representations of knowledge in the learner's head which can be compared, piece by piece, with the system's representations and that reasoning occurs by manipulating such representations. This

may be a computational convenience or it may reflect a profound philosophical view of the nature of knowledge. In the latter case, it has been challenged by recent work on situated cognition (e.g. Clancey, 1990). But this is a debate about the content (if any) of R_L*k - even the advocates of situated cognition emphasise that an agent (such as the system) must have representations of another agent's knowledge (R_SR_L*k) in order to communicate with that agent.

Our model integrates the discrimination of Wenger (1987) between 'behavioural diagnosis' (related to R_SR_Lbk in our model) and 'epistemic diagnosis' (related to R_SR_Lck in our model). However, the framework does not address what Wenger calls 'individual diagnosis'. The emphasis is on the dynamic modelling of learner knowledge states, not on modelling longer-term, knowledge-independent, individual attributes. The justification for this is not that such attributes are necessarily unimportant (although the designers of ITSs have not yet found great need for them) but that ITS research has yet to consider them in any detail. In particular, there are no significant techniques for inferring such attributes from learner's behaviour. If such attributes were to be added to the framework, then they would not be expressed in a differential fashion but as an extra dimension to the learner column.

Our distinction between bk and ck is related to that of Clancey (1986) between a 'situation-specific model' and a 'general model'. The general model "describes what is known about the world - for example, knowledge about stereotypic patients, diseases, and treatment plans"; the situation-specific model "includes the specific problem information, transformed or reorganized in some way, depending on the nature of the task" (Clancey, 1986, p395). Thus, the situation-specific model adopts (using some inference procedure) part of the general model to generate specific behaviour to solve a problem. Clancey considers in detail how the qualitative structure of these models and inference procedures differ in different domains. In comparison, our model is rather domain-independent in that we are not concerned directly with the contents of the various components of the framework for different domains but more with the general relationships between the components and how the various learner modelling techniques relate to these components.

Our model has to be simpler than the reality, as manifested in existing ITSs, since these are rather undisciplined agglomerations of miscellaneous techniques. Our model is only useful if it erases non-pertinent details in order to give a clear view of the modelling process. This means we will have to abandon irrelevant details or to "cut the corners" in our descriptions of some systems in order to integrate them neatly within the framework. Nonetheless, we feel that the framework captures the essential distinguishing features of the various techniques (as described in sections 3 and 4).

2. TERMINOLOGY

The terminology we outlined in the previous section has now to be elaborated. Much of the terminology of learner modelling has been introduced for a specific application or approach. This specificity reduces the ability to generalize across approach descriptors : if one approach X is described by characteristics A and B, it does not mean that A and B are dependent on each other, or specific to X, but that X results from a combination of A and B. We believe our discipline will progress if some general descriptors are available. Obviously, since these generalised descriptors lose the specificity that their authors introduced, there is a risk in proposing less specific descriptions. This will be corrected later, where approaches and systems are described individually. In the mean time, this common terminology will favour comparisons between approaches and ease the identification of the focus of various approaches. We aim for instance to show that some approaches are not incompatible but complementary, because the points on which they differ constitute different parts of the diagnosis process.

2.1 The meanings of "behaviour" and "solution"

Until now, we have used "behaviour" to describe the *solution* proposed to a problem P by the learner or by the system. For an equation 3x+5=20, the solution is x=5 ; for a medical diagnosis problem, the solution is the diagnosed disease, etc. The main characteristic of the learner's behaviour is that it is observable by the system : if the *learner's behaviour* $B_L P$ is defined as a sequence of actions the learner performs on the system interface then $B_L P = R_S B_L P$.

This definition is less clear when we consider the "product versus process" issue. Some ambiguity arises because some intermediate steps of the inference process also produce behaviour : for instance, the sequence of transformations written when solving an equation or the sequence of questions asked by the doctor. In AI literature, the word 'solution' is used for describing not only the final solution but also the solution process. Hence we prefer to call the sequence of actions which relate P to B (or in other words, some trace of the inference process) the *solution path*. The *solution* is usually the last step of the solution path. This definition does not eliminate the ambiguity completely, for instance in theorem proving, where the solution process (the sequence of logical transformations) is indeed the expected solution.

When some steps of the solution path are made observable to the system, they will be considered to be a part of the behaviour. Hence, the *learner's behaviour* may be defined as the observable subset of the solution path, generally including the last step.

Our definition of a learner's behaviour emphasizes the mental/observable characteristics of inference steps. This distinction does not hold for systems since all of the system's inference steps from P to B may be hidden or made observable by the designer. Subsequently, we will focus on the subset of the *system's behaviour* $B_S P$ which corresponds (in a sense to be discussed) to the learner's behaviour.

We will also see that, in many cases, learner modelling will not be performed from a single behaviour but from a set of behaviours $\{B_L P\}$ produced for a set of problems $\{P\}$. We will see later on the rationale for enlarging the behaviour to a larger subset of the inference process or to several behaviours.

2.2 Classes of discrepancies

The ITS literature has given different names to the various Δ relationships (discrepancies between the model's columns), as shown in Figure 4:

2.2.1 System - (represented) learner discrepancies

The first kinds of discrepancy are between the system's knowledge and the (represented) learner knowledge. These discrepancies have been (inconsistently) called misconceptions, bugs, mal-rules, etc. We try here to propose some stricter definitions. These discrepancies are the key objects in the learner modelling process which has mainly been approached in a *differential* way, i.e. by characterizing the learner's knowledge by its differences with respect to the system's knowledge.

< about here: Figure 4. Misconceptions, bugs and errors >

A *misconception* refers to a discrepancy at the conceptual level. The real misconception is the discrepancy between the system's and the learner's representation of ck. Since the system has no direct access to the learner's representation of ck, misconceptions are approximated by the discrepancy between the system's ck and the system's representation of the learner's ck :

$$\Delta(R_S ck, R_L ck) \approx \Delta(R_S ck, R_S R_L ck)$$

A *bug* refers to a discrepancy at the behavioural level. By contrast to the psychological connotation of the word "misconception", the term "bug" comes from the language of computing: in the early work of Brown and Burton (1978), discrepancies were represented by buggy procedures. Since the system has no direct access to the learner's representation of bk, bugs approximate the relation between the learner's and the system's representations of bk:

$$\Delta(R_S bk, R_L bk) \approx \Delta(R_S bk, R_S R_L bk)$$

Since the behavioural model has generally been represented as a production system, bugs have also been called *mal-rules*. The terms "*bug catalogue* " or "bug library" simply refer to a set of bugs, generally predefined by the system designers.

An *error* refers to a discrepancy between the learner's behaviour and the system's behaviour. Here, most systems are based on the postulate that the system has a perfect representation of the learner's behaviour. This postulate holds if, as above, we define the learner's behaviour as the set of actions he or she performs on the system's interface (and exclude any mental or off-system activity). In this case:

$$B_L P = R_S B_L P \iff \Delta(B_S P, B_L P) = \Delta(B_S P, R_S B_L P)$$

This definition assumes that there is only a single system's behaviour which is deemed to be correct. In general, this is not the case. The $R_S bk$ may not be fully deterministic; for example, it may represent a procedure for manipulating algebraic equations in which transformations may be carried out in different orders and yet eventually lead to an acceptable solution. In this case, we might define a learner to be in error if $R_S B_L P$ differs from all such $B_S P$s, and the error to refer to the discrepancy between the learner's behaviour and the 'best matching' system's behaviour (although this clearly gives rise to difficult diagnostic problems).

The term *diagnosis* (sometimes "*cognitive diagnosis*") refers to the interpretation of an error, as performed by the system. According to the designer's focus on the conceptual or behavioural level, the outcome of a diagnosis should be the identification of one or more bugs or misconceptions. The slightly different nuances associated with the terms "diagnosis" and "learner model" might derive from the fact that the latter implies that one aims to represent the learner's mental model while the former would represent a more narrow and faulty piece of knowledge. We do not retain these nuances and use the term "(cognitive) diagnosis process" synonymously with "learner modelling process".

An "*overlay model*" (Carr and Goldstein, 1977) is one in which the only possible diagnosed discrepancies are missing pieces of knowledge: $R_S{*}k \supset R_S R_L{*}k$. The use of the term "overlay" is not ideal because of its ambiguity: "partial model" would definitely be better, but “overlay” has already a long history in ITS.

The term "*perturbation model*" specifies that $R_S R_L bk$ may include elements which are not part of $R_S bk$ and hence ‘perturb’ its functioning.This term emphasizes the atomicity of these discrepancies. This atomicity presents computational advantages (it eases the search problem, discussed later) but appears to be poorly plausible at the psychological level as soon as one leaves highly-constrained domains.

We do not like the negative connotations of terms such as bugs or mal-rules. Rather than speak about "buggy" learners, we prefer to search for a rational account of why learners develop cognitive processes which are ill-adapted to Ω. In many cases, the

"buggy" process may be the result of an intelligent adaptation to previous situations, successful when acquired but now inadequate in Ω. Repair theory (see section 3.2.2) opens the way in this direction. However, we will continue to use the term "bug" because it is now standard in the ITS community.

The same nuance applies with conceptual discrepancies: the term "misconception" is negatively connotated. In particular, it neglects the fact that a learner may adopt a *viewpoint* which, although different to that of the system, is still reasonable. The difference between a misconception and a viewpoint is double. Firstly, a viewpoint emphasizes the fact that various approaches to the same domain may be pertinent, and that there is not necessarily one approach which is more valid than the others. Secondly, the term "viewpoint" indicates some conceptualisation of Ω which is larger than a misconception and is, in some way, closer to the idea of mental model (see section 3.1.2).

2.2.2 Learner - represented learner discrepancies

$\Delta(B_L P, R_S B_L P)$ represents a source of *noise* at the input of the learner modelling process. As said in the previous section, this discrepancy is usually considered to be non-existent. In general, this discrepancy denotes any misrepresentation of the learner's actual behaviour.

$\Delta(R_L{*}k, R_S R_L{*}k)$ represents the discrepancy at the output of the learner modelling process. We call it the *diagnosis error*. The diagnosis error indicates how precisely the represented knowledge approximates the actual knowledge possessed by the learner. Since the 'sum' of the discrepancies around the $R_L{*}k, R_S R_L{*}k, R_S{*}k$ triangle must be zero:

$$\text{diagnosis error} = \Delta(R_L{*}k, R_S R_L{*}k)$$
$$= \Delta(\Delta(R_S{*}k, R_L{*}k), \Delta(R_S{*}k, R_S R_L{*}k))$$

A *misdiagnosis* is defined to be a diagnosis error which is considered to be significant or important (as discussed in section 2.5.1).

2.3 Classes of consistencies

Several kinds of consistency relationship may be located in the model :

$\Phi(R_S bk, B_S P)$ expresses the consistency between the system's behavioural model and its behaviour. It is determined by the logical theory included in the behavioural model. If $R_S bk$ is a production system, $\Phi(R_S bk, B_S P)$ is determined by the logic of the associated inference engine. The same comment is valid for the consistency of the eventual direct relationship between the conceptual knowledge and the behaviour: $\Phi(R_S ck, B_S P)$.

$\phi(R_S ck, R_S bk)$ expresses the consistency between the system's conceptual and behavioural models of the same domain. When these both exist (which is seldom), these models are usually implemented separately. This means that this kind of consistency results from the competence of the knowledge engineer who performed the *knowledge acquisition* process.

$\phi(R_L bk, B_L P)$ is affected by the main source of noise in learner modelling: many behaviours may not be related to characteristics of the behavioural knowledge but to factors such as distraction, tiredness, and involuntary mistakes. These are called *non-systematic mistakes* or *slips*. (Slips are specific to human agents, not computer systems.) A slip is defined to be a piece of behaviour which does not correspond to the agent's intention. A slip, as opposed to a mistake, will generally be recognised as such if it is pointed out to the agent. Assuming that a slip occurs only intermittently, it may sometimes be detected by a difference between two behaviours produced by the same agent as a solution for the same problem. One generally does not ask a learner to solve the identical problem twice but instead to solve problems considered to be equivalent. Two problems P_i and P_j are considered to be equivalent ($P_i \approx P_j$) if their solutions require the activation of the same pieces of behavioural or conceptual knowledge:

$$slip = \Delta(B_L P_i, B_L P_j) \mid P_i \approx P_j \; ; \; P_i, P_j \in P$$

This definition shows the complexity of identifying slips. Since P_i is always slightly different from P_j, the behavioural difference may be related to a very context-specific element instead of due to some slip. The fact that some errors may be related to only a very small subset of P is not a kind of noise, it is a characteristic of the learner knowledge. Accepting that the learner's representation of the task includes some very task-specific elements is necessary if we want to describe the complexity of human behaviour with the seductive image of mental models. Moreover, P_i and P_j are presented at different times. Hence, variations between the respective behaviours may be the result of real knowledge changes, e.g. the result of learning or forgetting.

$\phi(R_S R_L bk, R_S B_L P)$ is a crucial parameter of the quality of the diagnosis process. In general the logical theory used by $R_S bk$ is also used by $R_S R_L bk$ since the latter is often viewed as a variation of the former. However, the postulate that the learner uses a reasoning process isomorphic to the system's logical theory is difficult to hold. Isomorphic does not mean that the processes are themselves similar but that they produce similar effects. It is here that the issue of psychological validity takes its main importance.

The relations $\phi(R_L ck, R_L bk)$ and $\phi(R_S R_L ck, R_S B_L P)$ are still more complex. They refer to the process of knowledge compilation, discussed by Anderson (1983). The relation between conceptual knowledge and behavioural knowledge is a matter of great controversy, particularly at the psychological level (i.e. concerning $\phi(R_L ck, R_L bk)$).

Strictly, the controversy concerns mainly the left most column of our framework (Figure 4), since in computational knowledge representations (the middle and right most columns) the distinction is more or less standard, although still of course a matter of debate. For the moment, the distinction is adopted as a heuristic device for analysing existing learner modelling techniques.

2.4 Diagnosis space

The learner modelling process may be viewed as a search process. The search space is the set of $R_S R_L{*}k$ that the system is able to build. Usually $R_S R_L{*}k$ is built by analysing $R_S{*}k$ and $R_S B_L P$. Since the discrepancy between $R_S{*}k$ and $R_S R_L{*}k$ is defined as a bug or misconception, the size of the search space may be defined by all the products of applying some combination of the bugs and misconceptions to the system's knowledge (which does not mean that this is the way this space is searched). We will call the search space the *diagnosis space* and denote it Ψ.

At the behavioural level, the diagnosis space will be defined as :

$$\Psi_{bk} = \{ R_S R_L bk \mid R_S R_L bk = R_S bk \ ** \ \{bug_1 \ldots bug_n\}\}$$

The meaning of the ** operator will be defined later when discriminating between several approaches. Similarly, for the diagnosis space at the conceptual level we define:

$$\Psi_{ck} = \{ R_S R_L ck \mid R_S R_L ck = R_S ck \ *** \ \{mis_1 \ldots mis_n\}\}$$

< about here: Figure 5. Diagnosis spaces (Ψ) >

Obviously, the search in the diagnosis space may be pruned by the fact that the system's representation of the learner should be consistent with the learner's behaviour. Generally, a perfect consistency ($\phi(R_S R_L bk, R_S B_L P) = 1$) may not be expected, as we have to take some noise into account. If we denote the noise by N, which lies between 0 and 1, we may write:

$$\Psi_{bk} = \{ R_S R_L bk \mid R_S R_L bk = R_S bk \ ** \ \{bug_1 \ldots bug_n\}\}$$
$$\text{and } \phi(R_S R_L bk, R_S B_L P) = (1 - N) \}$$

As we will see, information about the learner's behaviour may be used in various ways (e.g. as an heuristic, as feedback).

2.5 Uses of cognitive diagnosis

A diagnosis process is a learning process, since it involves repeatedly proposing hypotheses and modifying them in the light of experience. The goal of learning, i.e. the usefulness of the acquired knowledge, is (as emphasised in work on explanation-based

learning) an important source of background and heuristic knowledge for a learning process. Therefore it is important to consider the reasons why cognitive diagnosis is to be carried out. The goal of the diagnosis process is to provide information for choosing among didactic alternatives, with the underlying assumption that the didactic choices are best made by the system. This has two consequences - a consideration of the pragmatics of learner modelling and of the time scale over which the results of learner modelling are to be used.

2.5.1 Pragmatic approach

We have assumed that the goal of learner modelling is to minimise the difference between the learner's knowledge and its representation by the system : $\Delta(R_L{*}k, R_S R_L{*}k)$. One may - rightly - object that the goal of learner modelling is that it provide information to be used by the tutoring component and that fine perceptions of the learner's knowledge are not useful if they are not required to make a choice between didactic alternatives. In other words, a *pragmatic approach* may be expressed by the following rule: if the diagnosis error is smaller than the difference between two didactic choices leading to the same goal, then ignore the diagnosis error (since it cannot produce the 'wrong' choice), otherwise try to refine the diagnosis.

Or, more formally: define a *didactic action* DA by a triple $({*}k_x, DA, {*}k_y)$, where $\Delta({*}k_x, {*}k_y)$ describes the expected knowledge changes associated with DA. This assumes a means-ends view of teaching where DAs are selected according to the state-goal differences. Assume that the system believes the learner to be in state $X = R_S R_L{*}k$. Then the pragmatic approach is:

if $\exists\ (X, DA_1, Z)$ such that $X = R_S R_L{*}k$
and $\forall\ (Y, DA_2, Z)$, $\Delta(R_L{*}k, R_S R_L{*}k) < \Delta(X, Y)$
then select DA_1
else Goal = Reduce $(\Delta(R_L{*}k, R_S R_L{*}k))$

But the obvious problem is that the value of $\Delta(R_L{*}k, R_S R_L{*}k)$ (the diagnosis error) is not known by the system. Such a rule is therefore not a rule the system may apply but rather a rule for system design, given that the designer has some idea of the potential amplitude of diagnosis errors in his system.

2.5.2 Adaptive / evolutive discrimination

We can discriminate between two kinds of system on the basis of the time-scale over which the results of learner modelling are applied:

- *adaptive systems* : the diagnosis is only made for adapting the didactic choices to a particular learner, and it is abandoned for the next learner.

- *evolutive systems* : the diagnosis output (or some intermediate steps of the diagnosis process) are recorded and integrated within the system's knowledge in order to improve the diagnosis process for following learners.

This discrimination is pursued further in sections 3 and 4.

3. DIAGNOSIS APPROACHES

In this section we present a survey of the various approaches to learner modelling, expressed in terms of the framework presented above. We will illustrate the approaches by referring to ITSs described in the literature. We will not describe the systems completely but only their diagnosis processes and the things necessary to know to understand them.

3.1 The search space

The diagnosis space, as defined in section 2.4, is

$$\Psi_{bk} = \{ R_S R_L bk \mid R_S R_L bk = R_S bk \ ** \ \{bug_1 \ldots bug_n\}\}$$

and similarly for the conceptual level. It is thus composed by combining bugs and misconceptions with the system's representations of conceptual and behavioural knowledge. We now compare the various ways in which the diagnosis space is described and searched. Variations concerning the creation of the diagnosis space result from:

- The system's knowledge : the diagnosis space is created by variations of the system's knowledge R_S*k. It is obviously domain dependent, which makes comparisons difficult. However, we are interested in studying how the knowledge representation affects the definition of the search space.

- The bug and misconception catalogues : they define (some of) the variations of the system's knowledge . We are especially interested in studying how the bug catalogue $\{bug_1 \ldots bug_n\}$ is determined: is it predefined by the system's designer, is it acquired in some way through interaction with a learner or with an expert, or is it built dynamically by the system itself, or what?

- The generic operators : these define how R_S*k and the bug and misconception catalogues interact to create Ψ. These operators are represented by $**$ and $***$ in our formalism.

This discrimination is not as clear as it (maybe) appears here. For instance, some systems do not generate the diagnosis space from the system's knowledge but directly from the learner behaviour. We will be more precise further on. If the diagnosis space is

created from the system's knowledge then, since this "expert" model is domain dependent, we are not concerned with its content but rather with its structure and its knowledge representation scheme. More precisely, our interest concerns the relationship between the system's knowledge and the diagnosis space.

3.1.1 Generic model

When the diagnosis space is created from variations of a single model R_S*k - which is the most frequent case - we call the model a *generic model*. System models (behavioural or conceptual) gain generative power from their modularity. A set of relatively independent pieces of knowledge enables the generation of alternative sets by suppressing or substituting individual pieces. Modularity enables the system to access specific parts of knowledge, i.e. to build a finer diagnosis. This is at the same time an advantage and a disadvantage : quite often, the difference between the learner and the expert may not be represented by a difference in a small 'piece of knowledge'.

This modularity enables the calculation of the search space size (#(Ψ)) as a function of the number of pieces of knowledge in R_S*k. In an overlay model for instance, the search space is the set of all models obtained by suppressing some number of pieces from R_S*k :

$$\text{if } \#(R_S*k) = n \text{ then } \#(\Psi) = 2^n$$

The size of the diagnosis space is increased if external pieces of knowledge (bugs, misconceptions) may be combined with missing pieces. This size is then also function of the size of the bug or misconception catalogue (see section 3.2).

Three formalisms which have been used to give R_S*k this modularity are production systems, procedural networks and declarative theories:

EXAMPLE : WUSOR (Carr and Goldstein, 1977)

The paper which introduced the term 'overlay model' did so with respect to a set of 20 production rules to play the game of WUMPUS, e.g.:

L1: A warning in a cave implies that a danger exists in a neighbour.

L3: If a cave has a warning and all but one of its neighbours are known to be safe, then the danger is in the remaining neighbour.

The learner's knowledge is represented by associating a value (known, indeterminate, unknown) with each of these rules (in this case, then, #(Ψ) = 3^{20}).

EXAMPLE : BUGGY (Brown and Burton, 1978)

The BUGGY system represents domain knowledge (about subtraction) by a network of procedures, sub-procedures, etc., down to a set of primitive actions. The aim is to achieve a level of description which enables bugs to be associated with individual components of the procedural network. In this case the components are not independent, that is, one cannot simply delete any component (as

assumed in deriving the $\#(\Psi) = 2^n$ equation above), although Young and O'Shea (1981) in a production system representation of subtraction emphasise the ability to build learner models by deleting rules.

EXAMPLE : SCHOLAR (Carbonell, 1970)

SCHOLAR adopts a domain representation, a semantic network, which we can regard as 'declarative'. Carbonell points out that a learner model might be built by annotating nodes and links in the network.

3.1.2 Multiple models

Another approach is to put a set of models (indicated by the layers in Figure 6) at the disposal of the system and to make the diagnosis space equal this set :

$$\Psi = \{R^1_S*k,\ R^2_S*k,\ \ldots\ R^n_S*k\}$$

Obviously the size of the diagnosis space corresponds to the number of models:

$$\#(\Psi) = \#\{R^1_S*k,\ R^2_S*k,\ \ldots\ R^n_S*k\}.$$

< about here: Figure 6. Multiple models >

This corresponds to an interesting - but seldom-used - approach which accords with empirical evidence that a novice's knowledge may be a conceptualisation fundamentally different from the expert's one. Hence the former cannot be represented by small perturbations of the latter. Moreover, the learner may have a number of conceptualisations (viewpoints) which may be brought to bear on the problem.

3.1.2.1 Ordered (or genetic) multiple models. The set of models the system has at its disposal may represent some progression with respect to the correctness or the completeness of a learner's knowledge. This means that the last model is considered to be the goal state for the tutoring system. This for instance is the case with the qualitative / semi-quantitative / quantitative progression à la QUEST, or with the predefined ordered sets à la LISP TUTOR or INTEGRATION KID.

EXAMPLE : QUEST (Frederiksen and White, 1988)

QUEST provides a progression of models that starts with simple qualitative models of electricity and gradually introduces quantitative circuit theory. The approach is motivated by cognitive science research which shows that students often cannot apply basic (quantitative) laws to solve simple qualitative problems, and indeed their qualitative reasoning often shows fundamental misconceptions. As the student learns, so the learner model moves through the progression. However, Frederiksen and White recognise that expertise derives also from the ability to integrate models of different types, although QUEST does not address this issue.

EXAMPLE : LISP TUTOR (Reiser, Anderson and Farrell, 1985)

The $R_S$bk of the LISP TUTOR is a production system modelling an 'ideal student' rather than an expert. This rule set is actually a subset of the complete production system available to the LISP TUTOR, tailored to suit the learner's level of knowledge - as the student progresses, so another rule set is switched in. The rule sets are specified in advance of the learner using the system.

EXAMPLE : INTEGRATION KID (Chan and Baskin, 1990)

The INTEGRATION KID is an environment involving three agents: the learner, the tutor, and a computer-based learning companion, which is supposed to collaborate with the learner in mastering the domain. The companion's skill should advance at roughly the same rate as the learner's. To achieve this, the companion is represented by a pre-defined succession of discrete simulation programs which are subsets of the complete domain knowledge available to the tutor.

3.1.2.2 Unordered multiple models. In this case the system has at its disposal a set of models but there is no implied progression between the elements of the set. The elements provide alternative 'viewpoints' on the domain. The elements may be equally incorrect or equally correct.

Current research in cognitive psychology shows that individuals have distributed models, i.e. a collection of partial models (DiSessa,1986), which are activated according to the context. These context-related partial models correspond to the concept of a viewpoint. Their situatedness fits with the conception of expertise as the ability to adopt *multiple viewpoints* on the same class of problems (again, according to the context). The need for integrating multiple viewpoints within ITSs has recently received increasing support from the ITS research community.

EXAMPLE : KANT (Baker, 1990)

In some domains (e.g. music analysis), the assumption that $R_S$ck and $R_S$bk represent definitively correct representations is unsustainable. Instead, we might provide a set of 'viewpoints' representing possible (incomplete, uncertain) beliefs about the domain. In KANT, the emphasis is on the negotiative process between the learner, who may have adopted one such (or another) viewpoint, and the tutor. KANT's learner model is an overlay on the system's belief set, which is dynamically derived from a musical parser.

3.1.3 Multiple generic models

Obviously, the generic model approach and the multiple models approach are not incompatible. Diagnosis is likely to be better if the diagnosis is based on several generic models : each generic model typically defines a global approach to the domain and perturbations bring minor changes in order to come closer to the learner's knowledge.

EXAMPLE : Ruth's system (Ruth, 1976)

To analyse student's programs, Ruth provides a set of (actually, only two) templates describing possible strategies (e.g. the binary search and Newton's methods for finding zeroes of a function). After determining the best fit, the program then proceeded to analyse any errors in the student's program with respect to the selected strategy on the basis of known bugs and misconceptions.

In general terms, this approach is the same as 'case-based reasoning' in AI. We have a few basic models (cases) which correspond to the main conceptions of the domain, we select one (as in case-based reasoning), and then adapt it to fit the learner's behaviour. So far, recent research on case-based reasoning has not been applied to learner modelling, but it is likely to become a major research direction.

3.2 The bug and misconception catalogues

The bug catalogue (and the misconception catalogue, if it exists) is a key feature of most diagnosis processes. It encapsulates the experience of teachers and psychologists who have observed learners' mistakes in the domain over several years. Its structure derives from the system's knowledge representation : mal-rules for production systems, buggy procedures for procedural networks, false declarations for declarative theories,...

The size of the diagnosis space is determined by the size of the system's knowledge R_S*k and the size of the bug catalogue :

$$\text{if } \#(R_S*k)=n \text{ and } \#(\text{bug catalogue})=m \text{ then } \#(\Psi) = 2^{n+m}$$

This exponentially growing size emphasizes the crucial importance of heuristics in the search process. But the size is not really the critical point to discuss. There are more important factors to take into account such as how several bugs interact to produce errors or how the diagnosis space is explored.

The bug catalogue characteristics also depend on how it has been acquired. We review several methods below. Let's remember that if the updated bug catalogue is only used with the learner who showed these bugs, we have an adaptive system; if the discovered bugs update the bug catalogue which will be used for any learner using the system later on we have an evolutive system.

3.2.1 Predefined bug catalogue

In a predefined bug catalogue the designers have collected information (themselves, through protocol analysis, or from the literature) about the range of usual bugs or misconceptions that learners show in a domain. The main drawback is the cost of this work: it is very time-consuming and may not be reused for ITSs in other topics. Another drawback is that the range of possible diagnoses is restricted to those anticipated by the

designer (but we will see that this drawback also exists for other approaches, even if it is better hidden).

EXAMPLE : LISP TUTOR (Anderson and Skwarecki, 1986)

The LISP TUTOR has a bug catalogue of some 1200 rules which are buggy variants of the ideal model's rules. These have been accumulated after years of protocol analysis. (The earlier BUGGY system had similarly built a catalogue of about 100 buggy rules for subtraction by laborious protocol analysis).

EXAMPLE : WHY (Collins and Stevens, 1982)

By analysing tutorial protocols and asking tutors to comment on their strategies, Collins and Stevens identify a number of learner misconceptions (about meteorology, in this case). Diagnosis involves a complex interaction (in natural language, and hence not implementable) to map from surface errors to misconceptions. They also emphasise the role of multiple viewpoints (causal, temporal, functional, etc.).

3.2.2 Generated from the system's knowledge

In this approach, bugs are obtained by transforming pieces of knowledge taken from R_S*k. If $R_S$bk is expressed as rules it is relatively easy to generate changes which may correspond to some common learners' mistakes. A typical example is overgeneralisation which may be obtained by deleting a subset of the condition part of a rule.

EXAMPLE : ET (Fum, Giangrandi and Tasso, 1988)

ET is an ITS for language learning which uses a standard bug catalogue. However, if a bug is suspected which is not in the catalogue, it is dynamically generated. For example, if the learner persistently uses tense t1 instead of t2, then the rule for t1 is generalised (by removing some and-clauses and adding some or-clauses to the condition part) and the rule for t2 is specialised (conversely). Of course, the difficulty with such syntactic transformations is the potential combinatorial explosion.

The process of generating bugs may be improved when based upon a psychological theory, since not all transformations of R_S*k are equally plausible:

EXAMPLE : REPAIR (Brown and VanLehn, 1980)

The theory proposes that bugs arise from repairs (local patches) performed at impasses (where incomplete knowledge leaves the learner unable to proceed). The impasses are generated by deleting rules from the correct procedure. The repairs are based on psychologically-motivated principles: that they are small, domain-independent and impasse-independent. Even so, the generated repairs have to be filtered through (domain-dependent) critics to eliminate implausible bugs. REPAIR theory failed to generate most of the previously-observed bugs, but did generate some additional bugs observed later and predicted the phenomena of 'bug migration', whereby the bugs exhibited by one learner vary within the class of bugs caused by the same impasse.

If it is possible automatically to generate malrules and misconceptions, then it may also be possible to apply similar techniques to generate correct rules and conceptions. This provokes the idea that instead of modelling the learner entirely with respect to *pre-specified* domain knowledge R_S*k, the learner could be modelled with respect to *dynamically generated* knowledge. Since the latter knowledge will be incomplete and partly inaccurate (given the limitations of machine learning), the system would probably better function as a collaborative partner, offering advice and suggestions about the material and the learning process, than as a tutor leading the learner to target expertise (Gilmore and Self, 1988; Dillenbourg and Self, 1990). The potential benefits of such an approach are that it might (a) reduce the demands on the accuracy of learner modelling, (b) focus more on important metacognitive skills, and (c) give learners a better view of what the learning process should entail.

3.2.3 Acquired from explicit interaction

This approach consists of interacting with a learner or with an expert-teacher. The system may present a set of mistakes that has been empirically collected or that it generates. The user is invited to describe the bugs or misconceptions underlying these mistakes. The interaction will obviously be different for a learner and a teacher. Such a component of diagnosis systems is called a *diagnosis space editor*. A diagnosis space editor may be viewed as a tool for knowledge acquisition, similar to a tape recorder or notebook for knowledge engineers, but with a greater interactivity and with automatic integration of new knowledge.

3.2.3.1 Explicit interaction with the learner. The difficulties in obtaining directly from learners useful descriptions of bugs and misconceptions for inclusion in a bug or misconception catalogue have been summarised by Wenger (1987, p. 392): "Not only does the current state of the art set technical limitations on dialogues between systems and people, but people's account of their own actions and understanding can be rather incoherent and sometimes unreliable. Even if they are coherent and reliable - and the language can be processed - there remains the issue of understanding these self-reports in terms of the models that learners have of the domain, of themselves, and of the system." Nevertheless, according to Wenger, "Sleeman has found in interviews that even fairly young learners are able to speak about their own knowledge of algebraic manipulations."

We need to distinguish between off-line designer-learner interactions (which is a version of protocol analysis which may lead to pre-defined catalogues), off-line system-learner interactions (where the 'system' here is a knowledge acquisition tool, not an ITS) and on-line system-learner interactions (where the 'system' is the diagnostic component of

an ITS). The first two are evolutive approaches, the third adaptive (unless the outcomes are kept for other learners, in which case, it too is evolutive).

3.2.3.2 Explicit interaction with an expert-teacher. Here the interaction is necessarily off-line, i.e. not with an ITS, but with a knowledge engineer or knowledge acquisition tool. We might hope that interactions with experienced teachers will lead to insights more quickly than does a lengthy protocol analysis by ITS designers.

EXAMPLE : BELLOC (Chanier, Pengelly, Self and Twidale, 1990)

BELLOC is a tutoring system for second language learning which can also be used by teachers in a diagnostic mode. A special interface presents examples (e.g. "Quel est sa adresse?") and leads the user to specify (if possible) various rules, conceptions, explanations, similar examples, and counter-examples associated with the given example. These inputs are then integrated into a structured pedagogically-oriented network which is subsequently used with learners. Other modes enable learners and trainee-teachers to attempt similar exercises. These interactions provoke valuable reflections about the domain and also provide data which can, to some extent automatically, be incorporated in the system's R_S*k.

3.2.4 Acquired from implicit interaction with the learner

In several approaches, the system uses mechanisms which enable it to extract bugs through some analysis of the learners' behaviour. These mechanisms are described in the following section on the search process. The important thing to note here is that in some systems the discovered bugs are recorded to enlarge the bug catalogue.

If the discovered bugs are simply added to the bug catalogue, the size of the search space is naturally increased. However, we can imagine that when new bugs are integrated with old ones in a more complex process, some bugs become partially redundant, which reduces the increase of the search space and may even decrease it.

3.3 The generic operators

Previous estimates of the size of the diagnosis space were based on the assumption of the existence of a *universal generic operator* such that any subset of $R_S bk$ may be associated with any subset of the bug catalogue:

IF $\forall\ k_i,\ldots,k_j \in R_S bk$, $\forall\ bug_n,\ldots,bug_m \in$ Bug catalogue,
$\{\ \{\ k_i,\ldots k_j\}$ ** $\{bug_n,\ldots,bug_m\}\ \} \in \Psi$

THEN ** is a universal generic operator.

This universal operator creates very very large diagnosis spaces. It actually ignores the relations between pieces of $R_S bk$ and individual bugs. For instance, in general adding

a bug may make no sense if the corresponding correct piece of knowledge is not suppressed from $R_S bk$. In a production system based on monotonic logic, we cannot have two contradictory rules such as $P \Rightarrow Q$ and $P \Rightarrow \neg Q$. If the combinations of pieces of $R_S bk$ and individual bugs are constrained by such logical requirements, the diagnosis space is highly reduced since bugs are automatically associated with pieces of $R_S bk$ to delete. In the extreme, the pieces to delete are the result of bug selection and the diagnosis space is hence reduced to all the possible bug combinations:

IF $\forall\ bug_i \in$ Bug catalogue , $\forall\ k_j \in R_S bk$

IF $bug_i \Rightarrow \neg k_j$ THEN bug_i ** $k_j \notin \Psi$

THEN ** is an operator preserving logical consistency

EXAMPLE : LISP TUTOR (Reiser, Anderson and Farrell, 1985)

The LISP TUTOR includes a buggy rule for the merging of two lists in which the function 'list' is specified instead of the 'append' which appears in the corresponding ideal rule. If the former rule is included in the learner model then the latter rule would be excluded.

This logical consistency presents the advantage of reducing the diagnosis space. However, it means that we cannot represent the sometimes contradictory knowledge of learners : a learner may sometimes behave as if both P and $\neg P$ are apparently believed. We say "apparently" because in fact P and $\neg P$ are generally believed in different contexts so that no contradiction is seen by the learner. These different contexts may be viewed as hidden conditions. Some work attempts to use beliefs systems for representing such inconsistencies and for performing diagnosis in general (e.g. Mizoguchi, Ikeda and Kakusho, 1988; Huang, McCalla and Greer, 1990).

The ** operator implies some psychological theory describing how correct and incorrect knowledge may interact. The best example is again repair theory, where ** enables the deduction of a bug from some missing knowledge with respect to plausible repair mechanisms.

EXAMPLE : ODYSSEUS (Wilkins, Clancey and Buchanan, 1988)

ODYSSEUS is the learner modelling component of GUIDON (Clancey, 1987) and, like repair theory, it generates learner models mainly by deleting and replacing components of the system's knowledge representations, which in this case are mainly concerned with factual, rather than procedural, knowledge.

The most studied characteristic of ** is the number of bugs which may be combined within one $R_S R_L bk$. We touch here on the notion of a simple versus *compound bug*. A compound bug is the interaction of several bugs. If ** only accepts simple bugs, then the diagnosis space equals the size of the bug catalogue. If ** accepts the interaction of several bugs, it exponentially increases the search space.

That is only one problem resulting from accepting compound bugs. A more dramatic issue is that the bugs' effects interact in a complex way on the learner's

behaviour. Hence, the process of inferring the bug from the behaviour becomes more complex. At the extreme, the combination of several bugs may sometimes produce a correct answer!

4. THE SEARCH PROCESS

The search process aims to match data ($B_L P$) with a model ($R_S R_L{}^* k$), or inversely to match the model to the data. Diagnosis search processes are ranged between two extremes :

- purely *data-driven* approaches : the diagnosis is built from the learner behaviour, without reference to a predefined model.
- purely *model-driven* : weak search methods explore the diagnosis space, generate models and match the model predictions to learner behaviour.

The first method is not feasible except in very simple domains; the second raises combinatorial explosion problems. Both neglect the heuristic value of the other method: models are required for interpreting behavioural data, and behavioural data are required for pruning the search space of possible models. However, the heuristic value of behavioural data is reduced by the presence of noise. Most search methods are somewhere between these two extremes - we present them from data-driven to model-driven:

4.1 Direct inference approaches

A *direct inference approach* (see Figure 7) is based on the postulate that the bug (and misconception) may be inferred directly from the error.

The first step is to identify the difference between the learner's and system's behaviours $\Delta(R_S B_L P, B_S P)$. This step is not as simple as it appears. What is for instance the difference between 3 and 6? 3 may be half 6, less than 6, odd, prime, etc. The search space for this difference is actually infinite. Hence, this approach requires an *error catalogue* which restricts the search space by telling the system which differences must be considered. By contrast with the bug catalogue, the error catalogue is often implicit : the considered differences are encrypted in the behaviour comparison process.

The second step is to infer a bug from the identified error. The term "direct inference approach" indicates that there are some predefined links between errors and bugs. This may be represented by a set of pairs ($error_i$, bug_j). The bug may then be directly deduced from the error. The point we want to make here is that there is no search during the second stage. The search is limited to identifying, among a set of errors, the one which best characterises the learner's behaviour.

EXAMPLE : WEST (Burton and Brown, 1976)

WEST uses 'issue recognisers' to relate learner behaviour to a set of independent issues (e.g. failing to use parentheses). WEST uses an overlay learner model and thus only represents missing, not incorrect, knowledge.

< about here: Figure 7. Direct inference approaches >

EXAMPLE : TAPS (Hawkes and Derry, 1989)

TAPS compares the learner solution with the expert solution and classifies each deviation in terms of a bug catalogue. (In fact, their catalogue is called an 'error classification' although it seems to include both errors, i.e. matters concerning behaviour, (e.g. 'hesitancy'), and bugs, i.e. matters concerning behavioural knowledge (e.g. 'student constructs schema that is not on the correct solution path')).

EXAMPLE : TDTDT (Daelemans, 1988)

The links from errors (e.g. using 'dt' instead of 't' in a Dutch word) to bugs (e.g. 'using third person instead of second person') are the result of the system learning through progressive refinement, using techniques developed for learning heuristic rules in second generation expert systems.

As we said, the error catalogue is generally not explicit. It may also be defined in a synthetic way, one rule defining several error-bug links:

EXAMPLE : GUIDON (Clancey, 1987)

Here is for instance a diagnostic rule used by GUIDON :

T-rule 6.05

IF The learner's hypothesis does include values that can be concluded by this domain rule, as well as others, and
The hypothesis does not include values that can only be concluded by this domain rule,
Some other values concluded by this domain rule are missing from the hypothesis

THEN The belief that the domain rule was considered by the learner is -0.70

The error here defined is the incompleteness of the learner's behaviour (it is more precisely a class of errors). The related bug is the absence of the rule in the learner's model, i.e. this rule associates an "is incomplete" error with an "is incomplete" bug. The variable "rule" appearing in T-rule 6.05 may be instantiated by many rules of R_s*bk. This enables GUIDON to associate in a very synthetic way many errors with their respective bugs. This economy does not modify the one-to-one relationship between errors and bugs.*

An identical approach may be designed for the conceptual level, i.e. for linking errors with misconceptions. However, such links are even more difficult to determine than the error-bug links.

EXAMPLE : MENO II (Soloway, et al, 1983)

MENO-II associates errors (e.g. READ(X) placed at start of program) with bugs (e.g. 'put READ statements with the declarations') and then with misconceptions (e.g. 'READ is a kind of declaration') by using links from entries in a bug catalogue to nodes of a network representing misconceptions. Some bugs may be linked to more than one misconception.

The direct inference approach to learner modelling raises many criticisms because it is generally very difficult to associate an error with one and only one bug or misconception.

4.2 Horizontal extension of the behaviour

The value of $¢(B_LP,R_Lbk)$ is mainly affected by non-systematic factors (see section 2.4). This kind of noise has been defined as behavioural variations between equivalent problems. Consequently, if one considers the common characteristics of several behaviours, i.e. if one abandons the variations, one reduces the noise and hence increases the consistency :

$$¢(B_LP,R_Lbk) < ¢(\{B_{iL}P_i\},R_Lbk) \mid \{P_i\}=\{P_1,..,P_n\} \text{ and } P_1 \approx ... \approx P_n$$

where $\{B_{iL}P_i\}$ represents this set of behaviours. We call $\{B_{iL}P_i\}$ a horizontal extension of B_LP because each behaviour of $\{B_{iL}P_i\}$ is at the same level (see Figure 8).

< about here: Figure 8. Horizontal extension of the behaviour. >

4.2.1 Synthetic behaviour

One may envisage applying inductive techniques to develop a synthetic behaviour which generalises the behaviour set $\{B_{iL}P_i\}$ into a single (hypothetical) behaviour B_LP. This synthetic behaviour is close to the concept of a mental model as a set of decontextualised rules of actions. Expert-teachers are probably able to perform the generalisation involved in this inductive process. However, they use in this generalisation process an enormous amount of knowledge which is not available to the computerized tutor.

EXAMPLE : PROTO-TEG (Dillenbourg, 1990)

PROTO-TEG attempts to learn conditions under which a didactic strategy should be applied, on the basis of learner model characteristics during (successful and unsuccessful) applications of the strategy. Heuristic techniques are used to 'smoothe over' the exceptions in the instances, although it is difficult to induce behavioural similarities without explicit generalisation knowledge.

4.2.2 Multiple diagnosis feedback

Most existing learner modelling systems have exploited $\{B_{iL}P_i\}$ as a kind of multiple feedback : verifying $R_S R_L bk$ by running it on problems P_1 to P_n and matching the results with behaviours from $\{B_{iL}P_i\}$ and eliminating diagnoses which do not predict a sufficient percentage of these behaviours. (See also section 4.4 on diagnosis feedback.)

EXAMPLE : DEBUGGY (Burton, 1982)

In general, a single example of behaviour (e.g. a solution of a subtraction problem) cannot be linked with a single simple bug or compound bug. DEBUGGY uses a generate-and-test method (using simple bugs and a few common combinations) to produce an initial set of hypotheses. These hypotheses can then be used to predict responses for other problems which can be compared with actual responses in the data set.

4.2.3 Temporal issues

This section raises the issue of time. Time is implicitly represented in our framework by the indices associated with the problems, since these problems have to be presented successively.

4.2.3.1 Urgency of didactic choices. The first difficulty is that collecting several behaviours before building a diagnosis implies that this diagnosis - and the subsequent didactic decisions - is postponed. Wenger (1987, p.383) expresses this issue as "... the contradictory requirements of being at once sensitive enough to adapt the tutor's attitude without delay, and stable enough not to be easily disturbed by local variations in performance". The solution to this difficulty must take into account the time learners spend on each problem, the learner's resistance to the frustration associated with a failure, and the danger of installing faulty behaviours by repetition, and must balance these factors against the gain expected from more valid diagnoses. This problem is not unsolvable: explanation-based techniques include a generalisation stage based on a single example - but have severe requirements with respect to the available knowledge:

EXAMPLE: MORE (Costa and Urbano, 1990)

MORE represents $R_S ck$ as axioms in predicate logic and aims to interpret a single example of behaviour (e.g. the statement that 'Louis XIV wore a wig for fun') by using explanation-based learning methods to generate a decontextualised representation of the misconception. MORE is one of the few systems to focus on $R_S ck$ rather than $R_S bk$, which is a standard theorem prover.

4.2.3.2 Longitudinal consistency. The second difficulty is that the learners are changing. We have described behavioural variations over time as noise, provided these behaviours are related to the same knowledge state. However, in many cases, $B_L P_i$ at time i and

B_{LP_j} at time j may result from different mental models. In other words, longitudinal inconsistencies may reveal knowledge changes instead of noise.

Longitudinal consistency has generally been expressed by statistical parameters. The problem is precisely that statistical measures "scratch" the time dimension and ignore the variations in knowledge over time. For instance, imagine two hypotheses are found to explain ten instances of behaviour. Let b1 denote the behaviours explained by the first hypothesis and b2 those explained by the second. If the set of behaviours is "b2 b1 b2 b1 b1 b2 b1 b1 b2 b1", we can say that the first hypothesis is more plausible (60%). However, with the same 60/40 distribution, we can have a behaviour sequence "b1 b1 b1 b1 b1 b1 b2 b2 b2 b2" from which we may deduce that the second hypothesis is better for the learner's knowledge at the end of the sequence of exercises.

These knowledge changes should be an important aspect of diagnosis. Behaviour should not be diagnosed in isolation but in the context of an on-going teaching and learning process. The current learner model and our expectations about learning outcomes should enable us to restrict the search space for an updated learner model.

EXAMPLE: WUSOR III (Goldstein, 1979)

The R_S*k for WUSOR III includes 'genetic' links between nodes in a network. These links are supposed to capture the evolutionary nature of knowledge, so that the learner model (implemented as an overlay on R_S*k) can guide the system's search for an updated model. Thus the frontier of the learner model provides a focus for the diagnostic process.

However, a satisfactory integration of the system's diagnostic process with the learner's learning process has yet to be achieved. It demands a more valid psychological model of learning (as well as domain representation) than we are currently able to provide.

4.3 Vertical extensions of the behaviour

Extending the behaviour vertically means including in it a part of the solution path or intermediate representations. This extended behaviour is closer to R_Lbk since it is intermediate between the problem and the solution. If the distance between the model and the behaviour is shorter, the consistency $\Phi(B_LP, R_SR_Lbk)$ will be higher and, subsequently, the diagnosis error $\Delta(R_Lbk, R_SR_Lbk)$ will be reduced. The extreme case would be where the learner's behaviour at the interface makes learner knowledge directly observable.

4.3.1 Including solution paths in the behaviour

We first examine the case where the learner's behaviour includes part of the solution path, i.e. some steps of the solution path (see Figure 9).

4.3.1.1 Model tracing. The concept of *granularity* is used to describe the distance between two intermediate steps acted by the learner. This distance is the amount of reasoning (or the number of decisions) the learner performs between two successive steps. A *fine-grain* description of the learner's solution path requires the learner to perform each single step of the solution. The *model tracing approach* (Anderson, Boyle, Corbett and Lewis, 1990) aims to constrain the learner to adopt a behaviour whose granularity is as close as possible (ideally identical, according to the underlying psychological rationale) to the granularity of the inference process used by R_{sbk}. This enables the system to match each learner step with one of its own steps, i.e. to "trace" the learner's behaviour. In the systems designed by Anderson and his colleagues, the granularity is very fine. This approach raises many questions concerning the (arbitrary) definition of the intermediate steps and the educational disadvantages of increasing the constraints on the learner.

EXAMPLE: LISP TUTOR (Reiser, Anderson and Farrell, 1985)

The LISP TUTOR uses its R_{sbk} to predict a set of possible next keystrokes. If the actual keystroke matches one of these, the corresponding ideal rule is added to the learner model. If none match, a buggy rule is identified. The granularity is intended to enable a unique identification.

< about here: Figure 9. Vertical extension : including observed solution path >

4.3.1.2 Reconstructing the solution path. Typical examples of a behaviour extended to include the solution path have already been given : equations, theorem proving, medical or technical diagnosis, etc. In some cases, the solution path is reconstructed by the system from R_{sbk} and B_{LP} (see Figure 10). This reconstruction raises very difficult questions of psychological validity.

EXAMPLE: PIXIE (Sleeman, 1982)

PIXIE takes a learner's solution and uses rules and mal-rules to work backwards towards the problem statement, thus hypothesising the learner's intermediate steps. The search is constrained by a set of domain-dependent heuristics. Moreover, when an intermediate step cannot be determined, PIXIE hypothesises a new mal-rule (which could be added to the bug catalogue). However, as with other syntactic methods mentioned earlier, there is no cognitive framework to prevent the generation of psychologically implausible mal-rules.

EXAMPLE : TDTDT (Daelemans, 1988)

The system's solution path is extended by adding predefined buggy alternatives on each node in order to generate all possible behaviours.

Most of the work in this area has appealed to the psychological status of production systems as cognitive models (rather than as a means for supporting computational modularity, as discussed in section 3.1.1).

EXAMPLE : ACM (Langley and Ohlsson, 1984)

ACM reconstructs a solution path by an exhaustive search using a set of primitive operators constrained by 'psychological heuristics'. It then interprets this vertical extension as a horizontal one: a solution path is viewed as a sequence of operator applications. The same operator may have been applied several times, in various conditions. Induction on these various conditions enables ACM to decontextualise them and infer the conditions the learner has associated with an operator.

< about here: Figure 10. Vertical extension : including reconstructed solution path >

4.3.1.3 Hybrid approaches. The method of reconstruction can be combined with that of model tracing. The intermediate steps generated by $R_S bk$ and used by the system for model tracing can also be used to infer missing steps in the observed behaviour.

EXAMPLE : IMAGE (London and Clancey, 1982)

IMAGE uses a model tracing approach to predict the learner's likely next actions according to the hypothesised plan and, if this fails, attempts to reconstruct the learner's action using heuristics linking observed actions with known strategic concepts.

4.3.2 Including intermediate representations in the behaviour

Extending the behaviour with some part of the solution path may be insufficient because the solution path never covers the complete reasoning process. In the case of equation-solving, for instance, the solution path does not include the mental activities performed to decide what will be the next step. Identifying the learner's plans (as a part of R_L*k) is a major difficulty in learner modelling. One approach is to constrain learners to extend their behaviour vertically by asking them to make their plans explicit (see Figure 11). This extension is called an *intermediate representation*.

< about here: Figure 11: Intermediate representations >

EXAMPLE : EPIC (Twidale, 1989)

EPIC requires learners to specify their plans before they input lines of a logical proof. The plan is represented by a natural language-like template which must be instantiated to the problem at hand. These plans and sub-plans must then be annotated by the learner as she progresses with the proof. These annotations enable the system to maintain a learner model representing plans which could be determined from the proof itself only with great difficulty.

There is only a nuance between an intermediate step (in the solution path) and an intermediate representation. An intermediate step indicates some progression in the problem solving process. An intermediate representation reifies some knowledge applied during this process. Another difference is that intermediate steps tend to make behavioural knowledge concrete while intermediate representations may also reify conceptual knowledge.

EXAMPLE : BRIDGE (Bonar and Cunningham, 1988)

BRIDGE is intended to provide a set of intermediate representations to enable novice programmers to articulate program designs. Users move through three phases: building a 'natural language program'; transforming this into a 'plan program'; and defining a Pascal program. Diagnosis is eased since instead of reconstructing intentions from code (as in PROUST) or monitoring code-level input (as in LISP TUTOR), BRIDGE has explicit representations of the learner's goals.

It is interesting to note that the learner's reification of reasoning processes presents advantages not only for the diagnosis process. This constraint appears to have positive educational effects since the learner has to perform the metacognitive activity to bring into consciousness some parts of the reasoning process which were often implicit. For instance, with EPIC learners said that they discovered the importance of plans in their proof constructions, and with BRIDGE learners must reflect on the programming methodology which BRIDGE imposes.

EXAMPLE : GEOMETRY TUTOR (Anderson, Boyle and Yost, 1985)

The GEOMETRY TUTOR requires students to develop proofs using a proof graph (a form of intermediate representation). In a related experiment, Singley (1990) showed that the activity of 'goal-posting' generally improves problem-solving performance.

4.4 Diagnosis feedback

The scheduling of diagnosis feedback (that is, feedback to the system about its diagnosis, not 'didactic feedback' to the learner) enables a differentiation between data-driven and model-driven approaches. In a data-driven approach, behavioural information is intensively used in building the hypothesis. Consequently, the feedback arrives later and with a higher probability of being positive. In model-driven methods, the behavioural information only partially prunes the diagnosis space and hence leaves many concurrent hypotheses. Consequently, feedback is required sooner and has a higher probability of being negative, i.e. feedback is more important.

4.4.1 Behavioural simulation

Given that $R_S R_L bk$ is runnable, it can produce a simulation of the learner's behaviour, which we denote by $B_L P'$. Hence $¢(R_S R_L bk, B_L P') = 1$. If the simulation matches the learner's actual behaviour, the diagnosis is confirmed:

IF $\Delta(B_L P', B_L P) < Noise$

THEN $\Delta(R_L bk, R_S R_L bk)$ is acceptable.

If not, one has to check if the difference between the behaviour and its simulation is not a matter of noise. There are two ways of tackling this appearance of noise: first, by using multiple diagnosis feedback (section 4.2.2) and secondly, by 'coercing' the learner's behaviour. A coercion is an attempt to explain (away) slips in some rational way. For example, the DEBUGGY system used a set of coercions representing common performance slips in an attempt to eliminate slips from the data. Of course, it is somewhat self-contradictory to seek rational explanations for slips, as defined.

4.4.2 Didactic prediction

If the learner model is valid, it can be used to select a didactic action, i.e. to predict the efficiency of this action or to anticipate the learner's knowledge after the action. If the predicted changes happen, the diagnosis is confirmed. This approach is described in the pragmatic approach (section 2.5.1).

We defined a didactic action DA by a triple $(*k_x, DA, *k_y)$, where $\Delta(*k_x, *k_y)$ describes the expected knowledge changes associated with DA. Didactic prediction uses $*k_y$ for confirming the $*k_x$ diagnosis. Using the diagnostic function f (from section 1.3):

IF $f(B_L P_i) = (R_S R_L *k)_i$

and $f(B_L P_{i+1}) = (R_S R_L *k)_{i+1}$

and $\exists\ (*k_x, DA_n, *k_y) \mid *k_x = (R_S R_L *k)_i$ and $*k_y = (R_S R_L *k)_{i+1}$

THEN $¢(R_S R_L *k, B_L P_i) \approx 1 - noise$

and $\Delta((R_L *k)_i, (R_S R_L *k)_i)$ is acceptable.

This approach raises two problems. First, we need to link learner models and didactic actions with certainty, i.e. to predict knowledge changes resulting from didactic actions. Secondly, the knowledge change has also to be attested through the diagnosis process, which makes the difficulty recurrent.

EXAMPLE : PROTO-TEG (Dillenbourg, 1990)

The recursion problem has been avoided by defining only one kind of effect of didactic actions, their efficiency expressed as the rate of correctly identified quadrilaterals : (bk_x, DA_n, 80%). In this case bk was much closer to a class of behaviours than to a behavioural model.

4.4.3 Behavioural prediction

Instead of behavioural simulation, we can have behavioural prediction: an additional problem P_j (equivalent to the initial problem P_i) is generated and the subsequent learner behaviour is used to confirm the diagnosis:

IF $f(B_L P_i) = (R_S R_L * k)_i$
$P_i \in P$ and $P_j \in P$ and $(P_i \approx P_j)$
$\Delta(B_L P_j', B_L P_j) < Noise$
THEN $\Delta((R_L * k)_i, (R_S R_L * k)_i)$ is acceptable.

EXAMPLE : IDEBUGGY (Burton, 1982)

Each simple bug has an associated 'heuristic problem generator' which produces test problems for that bug. If diagnosis provides more than one bug which is consistent with observed behaviour, then a set of test problems is generated (which give different answers for the different bugs) using the heuristic problem generators for those bugs and these test problems are submitted to the learner.

Here, we come back to the temporal issue: the method presupposes that the learner's knowledge does not change between P_i and P_j. The interest is in the interactive aspect of this approach : P_j may be generated by the system to permit a more precise diagnosis, for instance for discriminating two hypotheses. Ideally, the discriminating problems should be generated by domain-independent mechanisms analysing the representations of the bugs, not from pre-specified problem generators associated with each bug. Here we are concerned with disambiguating two (or more) hypotheses that the system has about the learner, but of course the learner may in fact hold two hypotheses, not one. Therefore generating a test problem may help both the system and the learner. Similarly, a test problem (a counter-example) might be generated to reveal (to the learner) differences between the learner's hypothesis and the system's 'correct' hypothesis.

EXAMPLE : PG (Evertsz, 1989)

PG has a correct production system model for the domain of fraction subtraction. Given a learner model in the same form, PG generates a counter-example by reasoning about the abstract computational behaviour of the learner model.

4.4.4 Explicit interaction

This approach consists of asking the learner to confirm the diagnosis - it is seldom applicable, but very efficient when it is:

EXAMPLE : ACE (Sleeman and Hendley, 1979)

ACE reconstructs the learner's solution path from the partial one specified and then asks the learner to confirm this reconstruction. ACE cannot handle erroneous partial solutions nor engage in any dialogue if the learner does not confirm the reconstruction.

EXAMPLE : MACSYMA ADVISOR (Genesereth, 1982)

The ADVISOR attempts to infer misconceptions (mistaken beliefs about MACSYMA's operations) from the user's inputs and then queries the user about these beliefs (e.g. "Did you expect COEFF to return the coefficient of D6?"). Notice that the ADVISOR attempts to bypass the problem of identifying specific bugs by addressing deeper misconceptions directly.

4.5 Inexact diagnosis

At the beginning of this section we said that the aim of the search process is to match data with *a* model. As is now apparent, this is an unreasonable aim, which perhaps should be recognised from the outset. In general, the learner model will contain a number of components of which the system is more or less sure. A number of techniques for handling this problem are implicit in the foregoing discussion. In addition, there are a few systems which address this point explicitly:

EXAMPLE : IMPART (Elsom-Cook, 1988)

IMPART proposes a 'bounded user model' in which the learner model is represented by a set of upper and lower bounds on the possible states of the learner. The bounds specify a 'version space' in machine learning terminology but there is no implication that the system should aim to bring the bounds together: instead, it is assumed that tutorial actions will take account of such bounds.

EXAMPLE : TAPS (Hawkes and Derry, 1989)

The learner model of TAPS uses fuzzy terms ('very likely', 'possibly') to indicate the extent to which a component is in the learner model. The didactic procedures use similar terms. It is not clear how the terms are updated.

EXAMPLE : SCENT (Greer, Mark and McCalla, 1989)

SCENT makes use of the concept of the granularity (used in a somewhat different sense to that in section 4.3.1.1) of a knowledge representation. The idea is that a diagnosis may be made at a superficial (low-grain-size) level or a deep (high-grain-size) level, or somewhere in between. Therefore, a set of R_S**ks (describing the same knowledge but at different levels of detail, unlike viewpoints (section 3.1.2.2) which are also a set of* R_S**ks but which represent different knowledge) are provided to enable diagnosis to be carried out at the appropriate level.*

5. CONCLUSIONS

We have attempted to provide a computationally-oriented conceptual framework within which methods for learner modelling can be described. Most if not all of the techniques described in the literature can be encompassed in this framework. The aim of developing the framework is to help make it easier to assess the contributions that individual techniques make to the general problem of learner modelling, and to see where individual techniques 'fit' within the general schema.

Apart from the general outcome of providing a framework and a notation for thinking about learner modelling, two main conclusions follow from this review:

1. That many approaches to learner modelling, although often presented as if they were in competition with one another, are in fact complementary since they address different parts of the framework - this is made clear in the figures, where annotations appear in different places.
2. That most of the work on learner modelling has been concentrated on the lower half of our framework, that is, on the behaviour <-> behavioural knowledge mapping, with a relative neglect of the conceptual knowledge component.

As a corollary of this report, we can anticipate a number of future activities in learner modelling research:

1. To improve the framework and notation, the latter at the moment serving a descriptive rather than analytic function.
2. To integrate complementary learner modelling techniques, rather than focussing on a single approach.
3. To extend and develop techniques to address more conceptual issues.
4. To consider whether and how the framework can be modified if any of the fundamental implicit assumptions listed in section 1.4 are withdrawn.
5. To adopt the framework in order to situate new techniques with respect to each other and previous work, thus reducing reinventions and enabling quicker progress in the field.

Acknowledgements

This work has been partly funded by the European Economic Community under the DELTA programme and carried out within the NAT*LAB project. We are grateful to Nicolas Balacheff, Thierry Chanier, Melanie Hilario, Patrick Mendelsohn, Michael Pengelly, Peter Reimann, Daniel Schneider and Michael Twidale for their comments on earlier drafts and to the reviewers for their detailed suggestions.

References

Anderson, J.R. (1983). *The Architecture of Cognition*, Cambridge, Mass.: Harvard University Press.

Anderson, J.R., Boyle, C.F., Corbett, A.T. and Lewis, M.W. (1990). Cognitive modelling and intelligent tutoring, *Artificial Intelligence*, 42, 7-49.

Anderson, J.R., Boyle, C.F. and Yost, G. (1985). The geometry tutor, *Proc. of Ninth IJCAI*, Los Angeles.

Anderson, J.R. and Skwarecki, E. (1986). The automated tutoring of introductory programming, *CACM*, 29, 842-849.

Baker, M.J. (1990). Negotiated tutoring, Ph.D. thesis, Open University.

Bonar, J. and Cunningham, R. (1988). Intelligent tutoring with intermediate representations, *Proc. of ITS 88*, Montreal.

Brown, J.S. and Burton, R.R. (1978). Diagnostic models for procedural bugs in basic mathematical skills, *Cognitive Science*, 2, 155-191.

Brown, J.S. and VanLehn, K. (1980). Repair theory: a generative theory of bugs in procedural skills, *Cognitive Science*, 4, 379-426.

Burton, R.R. (1982). Diagnosing bugs in a simple procedural skill, in D.H. Sleeman and J.S. Brown (eds.), *Intelligent Tutoring Systems*, London: Academic Press.

Burton, R.R. and Brown, J.S. (1976). A tutoring and student modelling paradigm for gaming environments, *ACM SIGCSE Bulletin*, 8, 236-246.

Carbonell, J.R. (1970). AI in CAI: an artificial intelligence approach to computer-assisted instruction, *IEEE Trans. on Man-Machine*, 11, 190-202.

Carr, B. and Goldstein, I.P. (1977). Overlays: a theory of modelling for computer-aided instruction, AI Memo 406, MIT, Cambridge, Mass.

Chan, T.W. and Baskin, A.B. (1988). Learning companion systens, in C. Frasson and G. Gauthier (eds.), *Intelligent Tutoring Systems*, Norwood: Ablex.

Chanier, T., Pengelly, M., Self, J.A. and Twidale, M.B. (1990). BELLOC: an interface for characterising students' applicable rules in second language learning, to be presented at *Cognitiva 90*, Madrid.

Clancey, W.J. (1986). Qualitative student models, *Annual Review of Computer Science*, 1, 381-450.

Clancey, W.J. (1987). *Knowledge-Based Tutoring: the GUIDON Program*, Cambridge, Mass.: MIT Press.

Clancey, W.J. (1990). The frame of reference problem in the design of intelligent machines, in K. van Lehn and A. Newell (eds.), *Architectures for Intelligence*, Hillsdale, N.J.: Erlbaum.

Collins, A. and Stevens, A.L. (1982). Goals and strategies for inquiry teachers, in R. Glaser (ed.), *Advances in Instructional Psychology II*, Hillsdale, N.J.: Erlbaum.

Costa, E. and Urbano, P. (1990). Machine learning, explanation-based learning and intelligent tutoring systems, in E. Costa (ed.), *New Directions in Intelligent Tutoring Systems,* Amsterdam: Elsevier (to appear).

Daelemans, W. (1988). Learning heuristic diagnostic rules in an intelligent tutoring system, AI Memo 88-20, AI Lab, Vrije Universiteit, Brussels.

Dillenbourg, P. (1990). Designing a self-improving tutor: PROTO-TEG, *Instructional Science*, 18, 193-216.

Dillenbourg, P. and Self, J.A. (1990). Designing human-computer collaborative systems, to appear in C. O'Malley (ed.), *Computer Supported Collaborative Learning*, Amsterdam: Elsevier.

DiSessa, A. (1986). Models of computation, in D.A. Norman and S.W. Draper (eds.), *User Centered System Design*, Hillsdale, N.J.: Erlbaum.

Elsom-Cook, M. (1988). Guided discovery tutoring and bounded user modelling, in J.A. Self (ed.), *Artificial Intelligence and Human Learning*, London: Chapman and Hall.

Evertsz, R. (1989). Refining the student's procedural knowledge through abstract interpretations, in D. Bierman, J. Breuker and J. Sandberg (eds.), *Artificial Intelligence and Education*, Amsterdam: IOS.

Frederiksen, J.R. and White, B.Y. (1988). Intelligent learning environments for science education, *Proc. of ITS 88*, Montreal.

Fum, D., Giangrandi, P. and Tasso, C. (1988). ET: an intelligent tutor for foreign language teaching, *Proc. of ITS 88*, Montreal.

Genereseth, M.R. (1982). The role of plans in intelligent teaching systems, in D.H. Sleeman and J.S. Brown (eds.), *Intelligent Tutoring Systems*, London: Academic Press.

Gilmore, D.J. and Self, J.A. (1988). The application of machine learning to intelligent tutoring systems, in J.A. Self (ed.), *Artificial Intelligence and Human Learning*, London: Chapman and Hall.

Goldstein, I.P. (1979). The genetic graph: a representation for the evolution of procedural knowledge, *Int.J. Man-Machine Studies*, 11, 51-77.

Greer, J.E., Mark, M.A. and McCalla, G.I. (1989). Incorporating granularity-based recognition into SCENT, in D. Bierman, J. Breuker and J. Sandberg (eds.), *Artificial Intelligence and Education*, Amsterdam: IOS.

Hawkes, L.W. and Derry, S.J. (1989). Error diagnosis and fuzzy reasoning techniques for intelligent tutoring systems, *J. of AI in Education*, 1, 43-56.

Huang, X., McCalla, G.I. and Greer, J.E. (1990). Student model revision: evolution and revolution, *Proc. CSCSI/SCEIO Conference*, Ottawa.

Langley, P. and Ohlsson, S. (1984). Automated cognitive modelling, *Proc. of the National Conf. on AI*, Austin.

Mizoguchi, R., Ikeda, M. and Kakusho, O. (1988). An innovative framework for intelligent tutoring systems, in P. Ercoli and R. Lewis (eds.), *AI Tools in Education*, Amsterdam: North-Holland.

Reiser, B.J., Anderson, J.R. and Farrell, R.G. (1985). Dynamic student modelling in an intelligent tutor for LISP, *Proc. of Ninth IJCAI*, Los Angeles.

Ruth, G.R. (1976). Intelligent program analysis, *Artificial Intelligence*, 7, 65-85.

Singley, M.K. (1990). The reification of goal structures in a calculus tutor: effects on problem-solving performance, *Interactive Learning Environments*, 1, 102-123.

Sleeman, D.H. (1982). Inferring (mal) rules from pupils' protocols, *Proc. of the European Conf. on AI*, Orsay.

Sleeman, D.H. and Hendley, R.J. (1979). ACE: a system which analyses complex explanations, *Int. J. Man-Machine Studies*, 11, 125-144.

Soloway, E.M., Rubin, E., Woolf, B.P., Bonar, J. and Johnson, W.L. (1983). MENO-II: an AI-based programming tutor, *J. of Computer-Based Instruction*, 10, 20-34.

Twidale, M.B. (1989). Intermediate representations for student error diagnosis and support, in D. Bierman, J. Breuker and J. Sandberg (eds.), *Artificial Intelligence and Education*, Amsterdam: IOS.

Van de Velde, W. (1988). Learning from experience, doctoral dissertation, AI Lab, VUB, Brussels.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*, Los Altos: Morgan Kaufmann.

Wilkins, D.C., Clancey, W.J. and Buchanan, B.G. (1988). Using and evaluating differential modelling in intelligent tutoring and apprenticeship learning systems, in J. Psotka, L.D. Massey and S.A. Mutter (eds.), *Intelligent Tutoring Systems: Lessons Learned*, Hillsdale, N.J.: Erlbaum.