Purpose Based Learner Modelling

Xiaolin Niu
Department of Computer Science
University of Saskatchewan
57 Campus Drive Saskatoon Saskatchewan S7N 5A9, Canada
001-306-966-2076
xin978@mail.usask.ca

Supervisors: Dr. Gordon McCalla & Dr. Julita Vassileva

ABSTRACT

This paper presents a brief review of early learner modelling in Intelligent Tutoring Systems focusing particularly on procedural knowledge representations vs. declarative knowledge representations. It then tracks the paradigm shift from traditional learner modelling, which emphasises knowledge representation, to distributed learner modelling, which focuses on the modelling process. Learner models in distributed, multi-agent environments are created by different agents for different purposes, are often fragmented and local, and only make sense in the specific context of creation. My work, in particular, is aimed at defining a taxonomy of purposes and retrieving information relevant to a particular purpose "just-in-time" in order to assemble and integrate fragmented learner model information about a learner. Examples of my approach will be drawn from the I-Help system. Some open issues in this approach are also given in this paper, which motivate my further investigation and exploration.

Keywords

Learner modelling, purposes, I-Help, agent, procedural model, declarative model, centralized and decentralised learner models

1. INTRODUCTION

Intelligent Tutoring Systems (ITSs) are computer-based instructional systems that are aware of the knowledge of the domain, as well as what the learner knows. In the context of ITS, a learner model is a representation of the computer system's beliefs about the learner and is, therefore, an abstract representation of the learner in the system. Learner modelling is a common task that is performed in almost every ITS for several reasons.

- First, learner modelling motivates the creation of systems that are adaptive to each learner's interests, preferences and background knowledge in order to provide personalized instruction to a particular learner. Learner modelling is the means by which the ITS could individually adapt the learning experience to suit the learner's perceived needs. [47]

- Second, learner modelling also enables location of suitable collaborators for the learner, as well as facilitating collaboration and co-operation between learners by taking into account the users' goals, plans and knowledge about domains.
- Finally, by making the contents of learner models accessible to students, learner modelling can be used to promote learner reflection and therefore to contribute directly to learning [5].

VanLehn [50] also outlined several functions that learner model most commonly perform in ITS. They are advancement, offering unsolicited advice, generating problems and adapting explanations. In general, learner modelling is important within ITS and has been a key research area for many years.

Traditional ITS focused on a single global description of a learner. However, software systems currently are shifting to distributed and agent-based. In this kind of environment, the traditional single, complex learner model is replaced by learner model fragments, developed by the various software agents populating the environment for particular purposes [53, 54][34]. These fragments come from a range of sources (e.g. raw data, other agents) and are partial, dependent on the context in which they were created, and distributed. The task of learner modelling, therefore, is shifting from the collection at one place of as many data about a learner as possible to collecting on demand whatever learner information is available at this moment and interpreting it for a particular *purpose*. Thus, the main focus of learner modelling shifts from traditional representation issues such as consistency maintenance to issues such as determining what knowledge to retrieve for a given purpose, making sense of this knowledge in context, etc. Learner modelling in this kind of system is a "just in time" [26] process, invoked as a part of achieving a particular purpose and using information relevant to that purpose [6]. My work, in particular, is aimed at defining a taxonomy of purposes and retrieving learner information relevant to a particular purpose just-in-time in order to assemble and integrate fragmented learner model information. The aim of this paper is to review early work on learner modelling, which mainly focuses on knowledge representation, and to introduce my *purpose-based* distributed approach to multi-agent learner modelling.

The reminder of this paper is organized as follows. A brief review of the most important learner modelling techniques is given in Section 2. Section 2 also overviews the distributed approach to learner modelling and compares it with centralized approaches. Section 3 explains why I am concerned with the purposes of learner modelling and gives the structure of purposes. Conclusions are drawn in Section 4, where I also outline my planned future work.

2. LEARNER MODELING

Learner Modelling (LM), as a field of study, has resulted in significant amounts of theoretical work, as well as practical experience. Learner modelling can be defined as the process of gathering and maintaining relevant information in regarding the knowledge state of a learner. Knowledge representation is the key issue in traditional learner modelling. In this section, I examine the past accomplishments in learner modelling, focusing on how the knowledge is represented.

2.1 Procedural Model vs. Declarative Model

Most Intelligent Tutoring Systems (ITSs) represent domain knowledge as a procedural model or a declarative model. Procedural models represent domain knowledge by a network of procedures, sub-procedures, etc., down to a set of primitive actions. In this kind of model, the main knowledge to be communicated is procedural, i.e. knowledge about how to perform a task. The aim is to achieve a level of description, which enables learner performance to be associated with individual components of the procedural network directly. In this case, the components are not independent, that is, one cannot simply delete any components [14]. This kind of model has strong diagnostic capabilities. The most famous system based on this model is the BUGGY system [4].

Declarative models represent knowledge in the form of a set of facts and rules organized so that the machine can reason with them while the set of knowledge is inspectable by the user. Declarative representations are easier to be directly maintained by the user by adding and deleting facts and rules. Usually the organisation of declarative learner models is according to instructional goals, and the information stored pertains to the learner's mastery of these goals. Since what is stored is the learner's mastery, these models store evaluations of the learner's knowledge by using symbolic or numeric value [43]. This model records features in a descriptive way, which means it contains description or evaluation of the knowledge instead of the knowledge itself. In the following section, several systems will be reviewed based on these two dimensions.

2.2 Review of Procedural Models

In a procedural model, the system compares the learner's answer with an intermediate result of an action that an expert performs in order to create and update the learner model about the domain knowledge [52]. Most systems apply a rule-based approach to induce a procedural model.

2.2.1 Bug Models

To account for learners' misconceptions in simple procedural skills, Brown and Burton proposed "the BUGGY model" [7]. The goal of the BUGGY program is to build a diagnostic model of the learner: a model of internalized incorrect instructions or rules. In this model, the learner's errors are seen as symptoms of a "bug". A collection of likely errors and misconceptions for a given domain are collected manually from empirical studies of learners in the domain.

BUGGY can help arithmetic teachers recognise learner bugs. BUGGY selects a problem related to a given bug, then asks the teacher to solve it using this incorrect procedure. The DEBUGGY program reads a learner's test answers and proposes the bug or bugs that most likely caused the learner's errors. IDEBUGGY is an interactive version of DEBBUGGY that adds the capability of dynamically generating problems to help identify learner bugs [24]. In the BUGGY model, the observed data is represented using a network of procedures or sub-procedures (methods and their variants), to simulate the behaviour of a learner. This network represents some possible ways that a learner could attempt to solve subtraction problems. When a learner's problem is presented to DEBUGGY or IDEBUGGY, various combinations of good and buggy methods are tried to find a complete procedure that most closely matches the performance of the learner [24]. In this process, unlike BUGGY, DEBBUGY does not calculate the answers of co-occurring bugs in advance. Rather it uses a generate-and-test method to produce a set of diagnoses, find the answers that predict and test those answers against the learner's answers, and keep the best matching ones. DEBBUGY and BUGGY can be used as off-line diagnostic systems because they work with a predefined test with subtraction problems and the learner's answers as input. IDEBUGGY, on the other hand, performs an interactive diagnosis, where the system can choose a problem whose answer will help diagnosis the most.

The BUGGY system provides a detailed model of the subtraction process. However, although the model can be used to reproduce learner mistakes, it does not explain why the learner has made those mistakes. Thus, BUGGY does not provide a deep model of the knowledge involved in the learner subtraction processes, so it is not clear which knowledge the learner is missing. Moreover, the expert knowledge of the system is a black box; it cannot be used to justify or explain the reasoning of the expert or the learner and therefore cannot be used to generate explanations to teach the learner. Also, all knowledge in this system is pre-defined. The new bugs will be added to the system when they are discovered. The system has no ability to augment the knowledge directly from the interaction. Finally, the effectiveness of this modeling is limited by the completeness and preciseness of the collection of bugs. This is a very important limitation because the process of acquiring such a collection is time consuming and laborious [55].

2.2.2 Machine Learning Approaches

In the BUGGY system, the static nature of bug libraries makes it impossible to model unanticipated learner behaviours. To capture novel learner misconceptions, machine learning techniques have been studied. Many applications of machine learning techniques to the construction of procedural models have been tried, some of which focus on extending a bug library [48] and some attempt to infer a learner model by using instance induction technique [31].

2.2.2.1 PIXIE

PIXIE [48] selects one model that produces the learner's behaviour from a set of offline-generated correct and buggy models, then recognises and interprets learner behaviour. PIXIE systematically searches the space of possible models, using an incremental method to build the learner model that effectively limits the size of the search space. At each stage, the potential models are limited to models generated from the existing model by adding either a correct or buggy rule for the skill being tested, or perhaps a mal-rule corresponding to an earlier skill. The system also attempts to generate new mal-rules when the learner exhibits a problem that cannot be modelled using the mal-rules already in the bug library. The search space is constrained by a set of domain-dependent heuristics.

There are some disadvantages of this approach. First, there is no general-purpose filtering mechanism to be used to cut down the number of new mal-rules presented to the learner. Learners have to decide which new mal-rules are appropriate extensions for the bug library.

Second, although the model can explain the behaviour of a learner, it gives no indications as to why learners have this particular set of mal-rules, or why learners in general make some types of mistakes but not others [24].

2.2.2.2 ACM

In order to avoid the cost associated with hand-constructed bug libraries, Langley et al. [31] tried to use machine learning techniques to diagnose misconceptions by applying condition (or instance) induction. ACM (Augmented Cognitive Modeller system) is a kind of reconstructive model in which domain knowledge can be decomposed into the set of primitive operators and the set of conditions for their applicability. The assumption underlying this model is that learners never err when performing the operators, but they sometimes apply operators to wrong situations. This assumption is less constraining than a pre-specified library of bugs.

ACM consists of three phases. In the first one, the problem space is established; in the second (off-line) phase, a solution path is found. This solution path is used to explain the learner's observed behaviour. There are three inputs to this phase: problem space, the definition and solution of the problem, and the learner's answer. The solution path is searched by using a set of primitive operators constrained by 'psychological heuristics' rules instead of exhaustive search. In the last phase, machine learning techniques are used to induce the procedure that is capable of generating the path identified in the previous phase. The output of induction is a set of conditions, which predict when an operator will produce a state. The state lies on the solution path connecting the input for a given problem to the learner's solution. The conditions found by induction are then used to specialize the operators, and the result is a procedure that models the learner's unique problem solving behaviour.

By using induction, ACM can construct models that capture both correct and buggy knowledge. However, because the complete procedure that models the learner's behaviour is induced from the final answers, the operators must be enough to model many kinds of behaviour, both correct and incorrect, the potential search space is very large although some "psychological heuristics" have been used to limit the search. The only remedy to this is collecting large amounts of data on each learner or imposing further constraints on the search space. But this remedy requires finding such constraints by using the very human-intensive methods [2].

2.2.2.3 INSTRUCT

To deal with the computational complexity in ACM, another approach based on machine learning techniques is proposed: INSTRUCT (37) (Interactive Student modelling using techniques of procedure indUCtion from Traces). INSTRUCT tries to induce models of procedural skill by observing learners' performance and collecting information both implicitly and explicitly. Two well-known techniques are employed in this approach: reconstructive modelling and model tracing, at the same time avoiding their major problems.

The main cause of complexity of reconstructive approaches (for example, ACM) is that the model is inferred from small amount of information. This results in huge search spaces and is time-consuming. INSTRUCT exploits the techniques of machine learning for procedure induction from traces. This approach uses incremental, on-line techniques to trace the procedure's execution performed by the learners. In such a case, a learner's solution path is induced by following the learner step by step in a manner is similar to model tracing. This technique not only traces all or some of the actions performed by the learner, but also includes sequencing information. However, although model tracing can be very successful for inexperienced learners, knowledgeable learners can be very frustrated if they only can perform one primitive operation at a time. By realising this, INSTRUCT induces macro-operators to allow learners to combine several operators in a larger chunk and to perform them at once.

The on-line and incremental features make INSTRUCT more effective and adaptive because immediate feedback on learner actions can be used to tailor the interaction between system and learner. Also the macro-operators allow the system to follow the learner more naturally. It does not rely on bug libraries and enables the learner to solve the problems using more complex steps. However, INSTRUCT needs to match the learner's actions to those proposed by the domain expert, so it needs an expert module. It is clear that building a domain expert is time-consuming.

In summary, procedural representation cannot explain why this procedure is generated; therefore it is difficult to provide appropriate feedback to the learner. Also, the pre-defined procedural (bug) library increases the overhead of the system and the library construction is time-consuming and expensive. It is impossible to have internal propagation in procedural models since it is hard to find relation between each set of procedures in the system.

2.3 Review of Declarative Models

Both the weaknesses and strengths of procedural knowledge representations are derived from the fact that they are use-specific [1]. In some cases, a more generalized declarative knowledge representation may be desired. The declarative model is a representation of what the system knows and is usually contrasted with knowing how to use facts. Declarative models are sometimes probabilistic, and can be represented using Bayesian Belief Networks, which allow to propagate new evidences and to update the knowledge. Several techniques can be used to construct this model: using stereotypes and combinations of stereotype [45] [25], fuzzy logic, semantic network, etc.

2.3.1 SCHOLAR

The earliest example of a declarative learner model is used in SCHOLAR [8], the first ITS, whose goal was to communicate information, in this case about South American geography. This system adopts a domain representation, a semantic network, which we can regard as "declarative". In the semantic network, the nodes represents various concepts, such as countries and products, linked by various relationships, such as part-whole or generalization hierarchy. The links allow certain fundamental inference processes on the network. For instance, the system can conclude that Santiago is in South America because Santiago is in Chile and Chile is in South America.

Carbonell pointed out that the semantic net representation of the knowledge base used in this project was close to the internal knowledge structure of human, and he believed that the learner model might be built by annotating nodes and links in the network.

2.3.2 GRUNDY: Stereotype-based User Modelling System

A frequently employed user modelling method is the so-called "stereotype approach" [44][45]. Stereotypes contain typical characteristics of user groups in the application domain of the system. They enable the system to make a large number of plausible assumptions on the basis of a substantially smaller number of observations. If certain preconditions are met, a stereotype can be activated for a specific user, which means that the assumptions contained in the stereotype become assigned to the user. These assumptions can be overridden by specific observations. The resulting collection of assumptions forms the individual user model.

A stereotype-based system is GRUNDY, which is a system recommending novels to people to read. The stereotypical profile in Grundy contains information about different facets, such as motivations, interests, etc. and their corresponding values and ratings. The stereotypes are arranged in a directed acyclic graph, formed by partial order generalization hierarchy. That means, for example, that the class of people described by SPORTS-PERSON is a subclass of the class of people described by ANY-PERSON. To determine which stereotypes are applicable to a particular user, GRUNDY asks the user for a self- description at the start of a session. Individual phrases (for example, athletic or sports man) would trigger stereotypes that are likely to apply to that user.

Stereotypes are valuable tools in domains where user classification is possible and relevant and is a reasonable approach for judging the user's personality traits. However, for judging the user's level of expertise, this approach is inaccurate, since the user's knowledge in the intermediate levels may be different from the system's concept or from other users' knowledge. Also, this technique requires considerable user effort and this increases the overhead of using the system.

2.3.3 Bayesian Belief Networks (BBNs)

Declarative models can represent knowledge using both symbolic value and numerical techniques. In SCHOLAR and GRUNDY, for example, symbolic techniques were used. However, intelligent systems often need the ability to make decisions under uncertainty using the available evidence. In order to capture the uncertainty inherent in modelling users, numerical techniques have been applied in last few years: such as Bayesian Belief Networks (BBNs), fuzzy logic (FL), Dempster-Shafer theory of evidence (DST) or neural networks.

A Bayesian Belief Network (BBN) is a directed, acyclical graph in which the nodes correspond to variables and links correspond to probabilistic influence relationships. They provide a mathematically correct and semantically sound model for representing uncertainty that provides a means to show probabilistic relationships between random variables. The networks are constructed based on the observable events and goals within each phase of the model. The propagation of changes in probability values is possible on receipt of evidence [46]. The casual organisation in BBNs facilitates the analysis of action sequences, observations, consequences, and expected utility [39]. Therefore, BBNs have a great versatility and power, and they are now the most common representation scheme for probabilistic knowledge. They have been used for modelling learners' knowledge by representing relationships among concepts.

Jameson [23] classified systems according to how BBNs have been used. Several systems mainly use the diagnostic inference capabilities of BBNs. Among them, some systems emphases on interpreting evidence about user's knowledge of concepts, such as IPSOMETER [20], OLAE [33], POLA [11], HYDRIVE [36]; some of them are designed to recognise the plans of an agent, such as [42], [20], WIMP3 [9]. Other systems combine diagnostic inference and prediction inference together by using both upward and downward propagation. For example, EPI-UMOD [12], POKS [13] are used to infer that user knows a concept given the fact that he/she knows certain other concepts; PRACMA [22], Ppp [51], VISTA- □ [19] predicting the user's cognition and behaviour form a basis for the system's decisions and actions.

Therefore, BBNs can be applied to user/learner modelling in many different ways. However the nodes in BBNs have to be initialised with some prior belief. This is problematic, because, for example, there may be no way to obtain meaningful prior knowledge when the system is being deployed for the first time. In this case, BBNs often assign equal prior probabilities to all hypotheses and this is not practical because it cannot distinguish between a state of ignorance and a genuine belief about a variable. Assigning the conditional probabilities to a BBN requires a large knowledge engineering effort. Moreover, the inference techniques in BBNs are NP-hard, so the computation is more complex.

2.3.4 Fuzzy Logic-based

In order to deal with uncertainty management in user/learner modelling, fuzzy logic (FL) is used based on two quite different consideration [23]:

First, people often reason in terms of vague concepts when dealing with the imprecision, or vagueness, which is typical of natural uncertainty. Fuzzy logic techniques can be used to mimic this human style of reasoning by representing and reasoning with vague concepts.

Second, users may express explicit information about themselves vaguely when they supply this information to a system. For example, a user says: "I don't know too much about HTML". The reason for that perhaps is because he/she does not have enough knowledge about HTML, or perhaps because he/she is not willing to express his/her knowledge precisely. In this way, the user's vagueness leads to uncertainty in the system representation. The functions of FL techniques are well suited to the representation of such input. They can form useful parts of a solution and allow such uncertainty to be represented and processed even when other uncertainty management techniques are used as well.

There are a lot of systems which use FL to model vagueness. For example, KNOME [10] – substituting Fuzzy Rules for the laws of probability; the sales assistant [41] – maximal use of minimal user input; GEORGETTE [30] – fuzzy rules for user simulation; SYPROS [18] – student modelling with fuzzy expert system; IFTRED [17] – replacing fuzzy rules with linear equations; SHERLOCK□ [27] – investigating the utility of alternative propagation techniques.

Fuzzy logic techniques are relatively easy for designers and users to understand since users often use vague terms than precise probability values to supply information to the system. FL also can be combined with other numerical techniques to deal with the user's input efficiently.

In summary, declarative models offer a clear, modular representation, which is transparent to humans and an ability to propagate the necessary modifications throughout a user model whenever any change is made.

2.4 Generic User/Learner Modelling Systems

Generic user/learner modelling systems motivate to provide user/learner modelling functionality as shells or server products. The user/learner modelling shell systems forming part of the application offer reusability and modifiability of user/learner models. Major shell systems developed during last ten years include UMT [3], BGP-MS [28][40], TAGUS [38], etc. According to Kobsa [29], a user/learner modelling shell system should be expected to provide as many services as possible; to be able to express as many types of assumptions about users as possible at the same time; and to have strong inferential capabilities. The user/learner modelling server systems are centralized software components that offer their services to several applications in parallel [15]. Compared to embedded user modelling components, user/learner modelling servers seem to provide promising advantages: information is maintained and processed in a central or virtually integrated repository; user/learner information acquired by one application can be employed by other applications; it is more convenient to update user/learner modelling information and relieve clients from user/learner modelling tasks, etc.

2.5 Decentralised Learner Modelling Approach

Decentralised learner modelling consisting of a multitude of learner models developed and kept by a variety of software agents in the context of multi-agent environments was described by Vassileva et al. [53]. This evolved into a new approach called "Active" learner modelling [34], which was proposed as a distributed alternative to server-based approaches. It emphasises the activity and context of modelling, rather than on the global description. The model is regarded as a function used to compute relevant information about one or more learners, depending on the purpose of adaptation and the context in which the need of modelling arises. In this sense, "model" is a verb rather than a noun. This approach is targeted at a distributed, multi-agent based software environment in which learners form a learning community. Therefore in this kind of environment, there is no single monolithic learner model associated with each learner. Rather the learner models are fragmented and distributed throughout the system. Decentralised learner modelling focuses on the modelling processes, such as retrieval, integration, and interpretation of the fragmented models, rather than traditional knowledge representation. This approach is the opposite of the centralized user modeling approach discussed in section 2.4

The centralized approach allows for the integration of existing information sources about users/learners and enables access to information stored in user/learner models. The pre-constructed user/learner models are stored in centralized or virtual centralized repository. This is contrast with decentralised (distributed) approach, in which user/learner models can be stored anywhere – in a centralized or distributed database, or in files. Even though they may be *physically* stored in a centralized database (e.g. in I-Help, where all the models are stored in a ORACLE database), the user data is *virtually* distributed since only the agents who created the model knows how to access it. While the centralized approach aims to collect as many data about user/learner as possible, the decentralised approach focuses on the process of collecting and integrating information about the user/learner at *particular times* and with specific *purposes*. The former must includes additional backup mechanisms in case of breakdown and security mechanism to guarantee the privacy of the expertise data; however, the latter does not concern these too much since there is no central model to be protected.

The authors argue [34] that in mobile, distributed, multi-agent software system, the decentralised approach could be a more suitable solution for user/learner modelling. To demonstrate this, the decentralised approach needs to be put on a more formal basis, which allows seeing its generality. The goal of my work is to create a generic representation of *purposes* for learner modelling. This representation will be *procedural*. A library of purposes for learner modelling will be created, which will look somewhat like the procedure libraries in BUGGY. The purposes will be organized on different levels of generality and will be associated with routines for retrieval, integration and usage of the learner modelling data. These procedures will be retrieved and executed by distributed components (agents), as they are needed to compute user models just in time for the purpose at hand. In the next section I will describe in more detail the approach that will be used to create such purpose-libraries for distributed learner modelling – I will call it "purpose-based learner modelling". Purpose-based learner modelling is decentralised (distributed) and active, i.e. it emphasises the process of modelling and the context in which modelling is happening.

3. PURPOSE-BASED LEARNER MODELLING

Purpose-based learner modelling mainly focuses on modelling processes, such as retrieval, integration, and interpretation. *Purpose-based* means that the distributed and fragmented models can be computed just-in-time [26], for a particular *purpose*, using only the data required for that purpose. There are two advantages in this approach. First, focusing on purposes can speed computation dramatically. Because agents in system only maintain distributed partial models, a full integration of the information would be expensive, and sometimes would be impossible. However, it would be possible to integrate specific data, which is relevant to a specific purpose. Second, retrieval and integration of information at a moment of time, for a particular purpose would take into account the local context and can be linked directly to the adaptation needed; that is the model only makes sense in the context in which it is created, such as time, purpose, the agent who created it, the available sources, etc. Purpose-based learner modelling will be explained using examples from the I-Help system [16].

3.1 Domain – The I-Help System

I-Help [16] is a distributed multi-agent system that allows users to request, receive and give peer help synchronously and asynchronously. There are two I-Help components: private one-to-one peer help and public discussion forums. The former provides peer help and expert advice and assists open-ended interactions between two individuals, such as tutoring. The latter allows learners to ask and answer questions on a variety of topics, which is more suitable for discussion involving multiple problem solutions and many people. These two components are useful for providing help outside normal class time and suitable for different types of interactions. Learner modelling in this environment can serve several purposes: e.g. selection of a suitable helper for match making purposes; evaluation of the learners by the teachers; self-assessment by the learner; reflection by the leaner; knowledge management by the course designer, etc.

I-Help is built on multi-agent architecture. Each learner has their own personal agent in which not only are the owner's characteristics and information stored, but so is fragmented information of other learners, who have ever contacted the agent's owner before. For example, the helper and helpee will evaluate each other and the results will update the knowledge profiles and be saved by each of the personal agents after each help session. Therefore, there is no single learner model describing an individual learner. Rather, learner models are represented and stored in a distributed fashion and computed on as-needed basis [6]. The purpose-based learner modelling is derived from this as-needed basis. Various information types can be modelled in I-Help system: knowledge, interests, cognitive style, eagerness, helpfulness, interaction preference, opinions of peers and user actions. The following section will focus on the purpose hierarchy, which is used to represent the purposes and contexts of the I-Help system.

3.2 Purpose Hierarchy in I-Help

A *purpose* can be regarded as a "packet" of data and processes. When a purpose is proposed, its packet is added to the current program environment so that its processes have direct access to what they need to know, without having overhead by needing to access to the entire knowledge of the whole system. It remains to be seen how to fill the detail of this scheme and how well it will work. An important feature of purposes is that they can be organized into *hierarchies*. The system can thus be viewed at many levels, from the very general to the very specific; i.e. the purposes in the hierarchy are linked together with *abstraction*-refinement relations where movement from finer to coarser-grained purposes. Specific purposes inherit information and procedures from the more general purposes above them in the hierarchy. Similarly to classes in OOP, a specific purpose may disable or provide exceptions (redefine) the information / procedures defined in a more general purpose. The structure of the purpose hierarchy is similar to McCalla & Greer's *granularity* hierarchy [35], which is a hierarchical semantic-network-style knowledge representation scheme in which an entity may be considered at a refined, detailed level or at a more general, approximate level. The more general purposes are called *higher-level purposes*, and the more specific purposes are called *lower-level purposes*. Both of them might compute information that comes from the raw data (See *Figure 1*, the dotted arrow line indicate that there are more levels between them).

Raw data (denoted by $R_1 ... R_n$) can be retrieved from the environment. One source of the raw data is users themselves who provide some profile information before using the system. Another source is user data stored by some applications, such as login information stored in a database. Other raw data comes from peer assessment. This kind of data is stored by various agents in their learner models. Thus, the raw data can be viewed as simple learner models created by various agents or applications and stored in a distributed fashion, i.e. there are many different "snapshots" for one user taken by different agents with different purposes in different contexts. This user information can be reused for various purposes, different than the original purposes for which it was stored, when some specific learner modelling task arises.

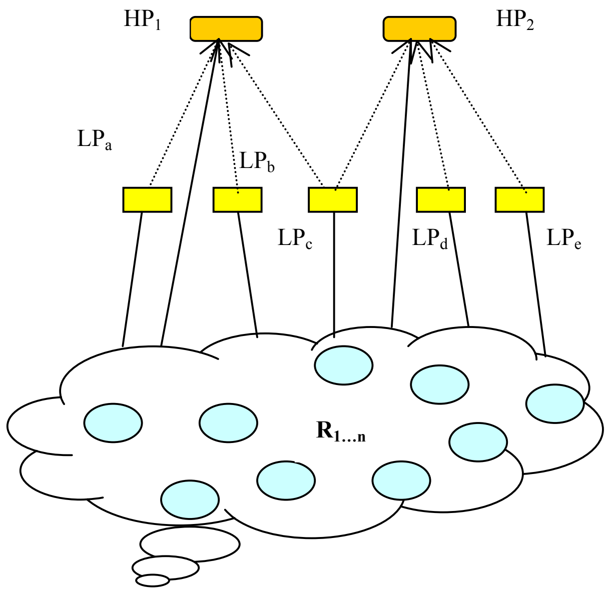

Figure 1. Purpose hierarchies in I-Help system

Inside each purpose, three kinds of information are stored: inputs, functions and outputs. The inputs denote the type of raw data from domain, which is relevant to the given purpose. The functions are sets of computational algorithms used to compute the inputs within the resource constraints by a particular computational agent in order to generate the desired outputs. The outputs are *partial* learner models, which are useful, reliable and appropriate for the purpose.

In *Figure1*, $LP_a...LP_z$ represents *lower-level purposes*. I have already developed some lower-level purposes for I-Help which concern different aspects of a learner, such as how knowledgeable is the learner; what is the learner's reputation; how active is the learner. For example, one *LP* might be how reputable learner *A* is. The inputs to this *LP* should include raw data about how many persons have given *A* premium and/or discount in trading help and how many persons A has been banned by. The functions inside this *LP* should be the production rules used to calculate the reputation of *A* by given such inputs. The output is the degrees of reputation of *A* rated by low, medium or high. The *higher-level purposes* (denoted by $HP_{1...}HP_n$) shift focuses from finer-grained to coarser-grained with fewer constraints. Some of them can serve for class statistics, or for open learner modelling purposes. For example, one such purpose can be that the teacher may want to compare the level of activity of the student with those of her/his peers in order to evaluate the student; or that a learner can ask the system how other people perceive him/her as a helper; or how does his/her activity in posting questions / answers relate to the level of activity of a given group, in order to reflect on his/her performance.

Partial learner models, as the output of various purposes, capture particular viewpoints of learners at various grain sizes. They are kept by each agent who has invoked this purpose, therefore are distributed and fragmented throughout the system. These partial learner models can be used as raw data, as inputs to other purposes.

Since the purpose hierarchy can be viewed as a structured representation of domain models arrayed with respect to *abstraction*, each lower-level purpose can be a specification of a higher-level purpose. For example, we can define HP_1 as comparing the characteristics of learner X to learner Y. The inputs to HP_1 can be various aspects about X and Y, such as knowledge level, reputation, activity, eagerness, helpfulness, etc. The functions are computational procedures: computing the value of each parameter for both X and Y; comparing the value of X and Y over each parameter, etc. The computation is performed on a very general level. The output is the result vector containing comparison result over each parameter. This higher-level purpose can be refined to some finer-grained lower-level purposes such as LP_a: comparing knowledge level of learner X with learner Y about topic T. The inputs to LP_a are the records of knowledge level for T of X and Y. The functions for LP_a can be simplified from $HP_{1:}$ comparing knowledge level for T of X to Y. Then the output will be generated by these functions. In summary, the purpose hierarchies capture the notion of context within a domain in which each purpose corresponds to a particular viewpoint from finer to coarser-grained on the domain. Moreover, such purpose hierarchies also prompt the automatic computing by re-using functions or procedures from other purposes.

3.3 Discussion: Some Issues about Purpose-based Approach

Purpose-based learner modelling is a complex process in which there are a lot of open questions (most of them also are questions for distributed learner modelling) involved, such as: How to define a purpose? How many purposes are enough? How to combine purposes? How general are those purposes? How useful are they? In other words, is the whole idea well defined and tractable? In this section, some issues which are not only related to purpose-based learner modelling, but also are common to the distributed learner modelling will be pointed out. Answering these questions still need further discussion and investigation.

The first question is if the model created for a specific purpose should be kept by each agent or not. If an agent keeps models of all other agents whom it has ever contacted, the number of models in the system will grow exponentially and the system's performance will become poor. It seems some models should be kept and some should be "forgotten". The question is what information should be kept, what should be thrown away. Perhaps integrating purpose-based learner modelling with reinforcement learning will help to filter out information that is relatively less context dependent (i.e. reusable, such as learner's characteristics profile, course and group information) and "forget" the strongly contextual information that can hardly be useful again. If some models should be "forgotten", when should they be thrown away?

Is it immediately after use or when they become a little bit “older”? Some of these decisions will probably depend on the purpose for which the model was created.

The second question is how to decide which information is relevant to a particular purpose or how one locates the agent that has a relevant model given the *purpose*. Vassileva et al. [54] suggest some criteria to be considered important when deciding which model to retrieve: why is the model needed (i.e. what is the current purpose for which information is needed); who has created the model; for what purpose the model was created; when was the model created and in what context was the model created.

The third question is how to make sense of possibly inconsistent and even contradictory data. The data relevant to a specific purpose comes from different learner models and perhaps is not consistent. For example, a purpose is to evaluate student A’s knowledge level. Several personal agents who have contacted A will be asked to provide information about A’s knowledge. However, the partial models about A’s knowledge stored by these agents are often inconsistent; for example, B is thinking A is intelligent while C is thinking A is foolish. There could be many reasons for this inconsistency: the models were created in different contexts and time; the relationships between the learners e.g. are they friends or enemies may play a role; the models were created based on different topics, etc. In this case, the question is how to or even whether to resolve the inconsistency?

The fourth question is how to interpret models created by other agents. The data stored in a learner model might be so specific that it only makes sense in the context in which the model was created. When a new agent comes and wants to re-use the model for its own purpose, the context may have changed. A similar problem occurs with providing letters of reference for a job candidate by various referees. In this case, the models of the candidate should contain only information about the requested features relevant to the job and created in a relevant context and time. The modelling agent will not make an attempt to make them consistent, but will create its own model based on an interpretation of these models. The function of the respective modelling purpose should provide for interpreting evidence from input from various sources taking into account the source reputation, etc.

4. CONCLUSION

This paper reviews traditional learner modelling techniques in ITS with emphasis on the differences between procedural versus declarative knowledge representation and centralized versus decentralised models. In a distributed multi-agent software system (such as I-Help), learner modelling is distributed and the focus shifts from knowledge representation to the modelling process consisting of retrieving, integration and interpretation. Learner modelling will be performed for a specific *purpose*, within a particular context.

The goal of the paper is to introduce a purpose-based approach for distributed active learner modelling. An important feature of purposes is that they can be organized into hierarchies. The system can thus be viewed at many levels of abstraction, from very general to very specific. The purpose-based approach not only has the potential to speed computing but also to capture the context of computation.

Some questions impact purposes directly such as how to define a purpose; how many purposes are enough; how general and useful are those purposes, etc. Some issues in the purpose-based distributed learner modelling still remain open for the moment, such as how to decide which information is relevant to a particular purpose; how to make sense of possibly inconsistent and even contradictory data; how to interpret models created by other agents and if the model created for a specific purpose should be kept by each agent or not.

In conclusion, a deep exploration of purposes and how they affect learner modelling will not only be useful to systems, for example I-Help, but will also shed light on distributed learner modelling issues. In particular, the creation of purpose taxonomy and constructing the purpose hierarchy will be the first step in showing how to make such distributed learner modelling both tractable and effective.

5. REFERENCES

[1] Anderson, J. (1988) The expert module, In Polson, M. C. & Richardson, J. J. (eds.), *Foundation of Intelligent Tutoring Systems*. Lawrence Erlbaum. Hillsdale, NJ. 21-54.

[2] Baffes, P., (1996) Refinement-based Student modeling and automated bug library construction, in *Journal of Artificial in Education*, 7, 1 pp 75-116.

[3] Brajnik, R. J., Tasso, C. (1992) A Flexible tool for Developing User Modeling Applications with Non-monotonic Reasoning Capabilities. *Proceedings of the third International Workshop on User Modeling. Dagstuhl*, Germany. 42-66.

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

[5] Bull, S. & Pain, H. (1995) "Did I say what I think I said, and do you agree with me?": Inspection and Questioning the Student Model, in J. Greer (ed), *Proceedings of World Conference on Artificial Intelligence in Education*. AACE, 501-508.

[6] Bull, S., Greer, J.E., McCalla, G., Kettel, L., Bowes, J. User Modelling in I-Help: What, Why, When and How. *Proceedings of 8th International Conference, UM 2001*, Sonthofen, Germany, July 2001, Pages 117-126

[7] Burton, R. R. (1982). Diagnosing Bugs in a simple procedural skill. In: D. H. Sleeman and J. S. Brown, (eds.): Intelligent Tutoring System. New York: Academic Press. [QUIL*: 116, 125; KASS*:394]

[8] Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer aided instruction. *IEEE Transactions on Man-Machine Systems*, 11, 190-202.

[9] Charniak, E., Goldman, R. (1993) A Bayesian Model of Plan Recognition. *Artificial Intelligence*, 64, 53-79.

[10] Chin, D. N. (1989) KNOME: Modeling What the User Knows in UC. InKobsa, A., Wahlster, W., (eds*), User Models in Dialog System*. Berlin. Springer. 74-107.

[11] Conati, C., VanLehn, K. (1996) POLA: A Student Modeling Framework for Probabilistic On-line Assessment of Problem Solving Performance. *In Proceedings of the Fifth International Conference on User Modeling*. Kailua-Kona, HI.

[12] de Rosis, F., Pizzutilo, S., Russo, A., Berry, D. C., & Molina, F. J. N. (1992). Modeling the user knowledge by belief networks. *User Modeling and User-Adapted Interaction*, 2, 367-388.

[13] Desmarais, M. C., Maluf, A., & Liu, J. (1995). User-expertise modeling with empirically derived probabilistic implication networks. *User Modeling and User-Adapted Interaction. Special Issue on Numerical Uncertainty Management in User and Student Modeling*, vol 5(3), 283-315.

[14] Dillenbourg P., Self J. (1992) A framework for learner modeling. Technical report AAI/AI-ED 74, Department of Computing, Lancaster University

[15] Fink, J., Kobsa, A. (2000): A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web. *User Modeling and User-Adapted Interaction* 10(3-4), Special Issue on Deployed User Modeling, 209-249.

[16] Greer, J.E., McCalla, G.I., Cooke, J., Li,ar. V.S., Bishop, A. & Vassileva, J., I. (1998). The Intelligent Helpdesk: Supporting Peer-Help in a University Course. *The International conference on Intelligent Tutoring Systems*. San Antonio, TX, USA, 494-503.

[17] Hawkes, L. W., Derry, S. J., and Rundensteiner, E. A. (1990) Individualized Tutoring Using on Intelligent Fuzzy Temporal Relational Database. *International Journal of Man-Machine Studies*. 33. 409-429.

[18] Herzog, C. (1994) Fuzzy techniques for understanding student solutions in intelligent tutoring systems. In R. Gunzenhauser, C. Mous, & d. Rosner (eds.), *Papers for the Seventh Meeting of GI Section 1.15/7.0.1, " Intelligent Tutoring Systems*". Ulm, Germany: Research Institute for Application-Oriented Knowledge Processing (FAW).

[19] Horvitz, E., Barry, M. (1995) Display of Information for Time-Critical Decision Making. In Besnard, P., and Hanks, S., (eds), *Proceedings of the Eleventh conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann. San Francisco, 296-314.

[20] Huber, M. J., Durfee, E. H., & Wellman, M. P. (1994). The automated mapping of plans for plan recognition. In R. Lopez de Mantaras & D. Poole (Eds.), *Proceedings of the Tenth Conference on Uncertainty in Artificial Inelligence* (pp. 344-351). San Francisco: Morgan Kaufmann.

[21] Jameson, A., (1992) Generalizing the Double Stereotype Approach: A Psychological Perspective. In Andre, E., Cohen, R., Graf, W. Kass, B., Paris, C., and Wahlster, W., (eds), *Proceedings of the Third International Workshop on User Modeling*. Dagstuhl, Germany, 69-83.

[22] Jameson, A., Schafer, R., Simons J., and Weis, T. (1995) Adaptive Provision of Evaluation-Oriented Information: Taks and Techniques. In Mellish, C. S., (ed), *Proceedings of the fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann. San Mateo, CA, 1886-1893.

[23] Jameson, A. (1996) Numerical Uncertainty Management in User and Student Modeling: An Overview of Systems and Issues. *User Modeling and User-Adaptive Interaction*. Special Issue on Numerical Uncertainty Management in User and Student Modeling, vol 5(3), pp 193-251.

[24] Kass Robert (1989) Student Modeling in Intelligent Tutoring System – Implication for User Mdoeling In: Kobsa, A. & Wahlster, W., (1989) (Eds.) User Models in Dialog Systems.

[25] Kay, J. (1994) Lies, damned lies and stereotypes: pragmatic approximations of users, *Invited keynote address in Proceedings of UM94 - 1994 User Modeling Conference*, UM Inc, Boston, USA, 1994, pp 175-184.

[26] Kay, J., (1999) A Scrutable user modeling shell for user-adapted interaction. Ph.D. Thesis. Baser Department of Computer Science. University of Sydney. Sydney. Australia.

[27] Katz, S., Lesgold, A., Eggan, G., and Gording, M. (1992) Modeling The Student in Sherlock . *Journal of Artificial Intelligence in Education, special issue on student modeling*, vol 3(4), pp 495-518.

[28] Kobsa, A., Pohl, W. (1995) The User Modeling Shell System BGP-MS. *User Modeling and User Adaptive Interaction. Special Issue on User Modeling Shell System*, vol 4(2), pp 59-106.

[29] Kobsa, A., (2001) Generic User Modeling Systems. In *User Modeling and User-Adapted Interaction* 11: 49-63, 2001.

[30] Kolln, M. E. (1995) Employing User Attitudes in Text Planning. *Proceedings of the fifth European Workshop on Natural Language Generation. Leiden, The Netherlands*, 163-179.

[31] Langley, P. and Ohlsson, S. (1984) Automated cognitive modeling*, Proceeding of the Second National Conference on AI, Austin*.

[32] Langley, P., Wogulis, J. and Ohlsson, S. (1990) Rules and principles in cognitive diagnosis, In Frederiksen, N., Glaser, R., Lesgold, A. and Shafto, M., editors, Diagnosis Monitoring of Skill and Knowledge Acquisition, chapter 10, pages 217-250. Hillsdale, NJ: Lawrence Erlbaum Associates.

[33] Martin, J. & VanLehn, K. (1995) Student assessment using Bayesian nets. Int. J. Human-Computer Studies (1995) 42, 575-591. Also available on-line: http://www.pitt.edu/~VanLehn/distrib/journal/HCS95.pdf.

[34] McCalla, G., Vassileva, J., Greer, J. and Bull, S. Active Learner Modeling. In *proceedings of The Fifth International Conference on Intelligent Tutoring System*, Montreal, 2000.

[35] McCalla, G.I and Greer, J.E. Granularity-Based Reasoning and Belief Revision in Student Models. In Student Models: The key to Individualized Educational Systems, J. Greer and G. McCalla (eds), New York: Springer Verlag, 1994, 39-62.

[36] Mislevy, R. J., Gitomer, D. H. (1996) The Role of Probability-Based Inference in an Intelligent Tutoring System. *User Modeling and User-Adaptive Interaction*. Special Issue on Numerical Uncertainty Management in User and Student Modeling, vol 5(3), 253-282.

[37] Mitrovic, A., Djordjevic-Kajan, S., Stoimenov, L., (1996) INSTRUCT: Modeling students by asking questions, *In User modeling and user-adaptive Interaction*, 6(4): pp 273-302.

[38] Paiva, A., Self, J. (1994) TAGUS: A User and Learner Modeling System. *Proceedings of World Conference on Artificial Intelligence on User Modeling. Hyannis*, MA 43-49.

[39] Pearl, J. (1997) Graphical Models for Probabilistic and Causal Reasoning in The Computer Science and Engineering Handbook, Editor Tucker, A., CRCPress, Boca Raton, FL, 1997, 699-711.

[40] Pohl, W., (1998) Logic-Based Representation and Reasoning for User Modeling Shell Systems. Sankt Augustin, Germany: infix.

[41] Popp, H., Lodel, D. (1996) Fuzzy Techniques and User Modeling in Sales Assistants. *User Modeling and User-Adaptive Interaction. Special issue on Numerical Uncertainty Management in User and Student Modeling*, vol 5(3), 349-370.

[42] Pynadath, D. V., V\Wellman, M. P. (1995) Accounting for Context I nPlan Recognition, with Application to Traffic Monitoring. In Besnard, P., and Hanks, S., (eds), *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann. San Francisco, 472-481.

[43] Ragnemalm, E. L. (1996) Student diagnosis in practice; bridging a gap. *User Modeling and User Adaptive Interaction. Special Issue on Special Issue on Student Modeling*, vol 5(2), pp 93-116

[44] Rich, E. (1979) User modeling via Stereotypes. *Cognitive Science*, 3: pp 329-354.

[45] Rich, E. (1989) Stereotype and User Modeling. In: A. Kobsa and W. Wahlster (eds.): User Models in Dialog Systems, pp. 35-51. Springer, Berlin, Heidelberg.

[46] P. Russell, S., and Norving, P. (1995) Artificial Intelligence: A modern Approach. Prentice Hall, New Jersey, 1995.

[47] Self, J. (1994) The role of student models in learning environments. *Technical report AAI/AI-ED 94*, Department of Computing, Lancaster University.

[48] Sleeman, D. H. (1982) Inferring (mal) rules from pupils' protocols, *Proceeding of the European Conference*. On AI, Orsay.

[49] Sleeman, D. H. (1985): UMFE: A user modeling front end subsystem. *International Journal of Man-Machine Studies* 23, 71-88.

[50][VanLehn, 1988] VanLehn, K. (1988) Student Modeling, In Polson, M. C. & Richardson, J. J. (eds.), Foundation of Intelligent Tutoring Systems. Lawrence Erlbaum. Hillsdale, NJ. 55-78.

[51][Van Mulken, 1996] Van Mulken, S. (1996) Reasoning about the User's Decoding of Presentations in an Intelligent Multimedia Presentation system. *In proceedings of the Fifth International Conference on User Modeling*. Kailua-Kona, HI.

[52] Vassileva, J. (1991) A classification and synthesis of student modeling techniques in intelligent computer-assisted instruction. In Norrie, D. H. and Six, H. W. (eds.), Computer Assisted Learning. *Lecture Notes in computer Science*. Springer-Verlag. Vol (438) 202-213.

[53] Vassileva, J.I., Greer, J.E., McCalla, G.I. (1999) "Openness and Disclosure in Multi-agent Learner Models", in Morales R. and Kay J. (Eds.) *Proceedings of the Workshop on Open, Interactive, and Other Overt Approaches to Learner Modelling*, International Conference on AI in Education, Lemans, France.

[54] Vassileva, J., McCalla, G., Greer, J. (to appear) Multi-Agent Multi-User Modeling. In *User Modeling and User Adapted Interaction.*

[55] Wenger, E. (1987) *Artificial Intelligence and Tutoring Systems. Computational and Cognitive Approaches to the Communication of knowledge*. Morgan-Kaufmann. Los Altos, California.