A Data-Driven Technique for Misconception Elicitation

Eduardo Guzmán, Ricardo Conejo, and Jaime Gálvez

Dpto. Lenguajes y Ciencias de la Computación. Universidad de Málaga,
Bulevar Louis Pasteur, 35. 29071 Málaga, Spain
{guzman,conejo,jgalvez}@lcc.uma.es

Abstract. When a quantitative student model is constructed, one of the first tasks to perform is to identify the domain concepts assessed. In general, this task is easily done by the domain experts. In addition, the model may include some misconceptions which are also identified by these experts. Identifying these misconceptions is a difficult task, however, and one which requires considerable previous experience with the students. In fact, sometimes it is difficult to relate these misconceptions to the elements in the knowledge diagnostic system which feeds the student model. In this paper we present a data-driven technique which aims to help elicit the domain misconceptions. It also aims to relate these misconceptions with the assessment activities (e.g. exercises, problems or test questions), which assess the subject in question.

Keywords: Student Modeling, Misconception Elicitation, Student Knowledge Diagnosis.

1 Introduction

Intelligent learning environments base their intelligence on the adaptive instruction they supply. The use of student models in such environments has emerged as a consequence of the fact that these systems have to work with incomplete information about the students [1]. A student model represents *who* is being taught, that is, what the student does (or not) know about the domain. Most learning environments construct this model from the student knowledge and the gaps in this knowledge. Using this information, they adapt the teaching process to the student's need. The quality of this adaption strongly depends on the accuracy of the student model. However, inferring the student model (and, in general, any user model) is a very difficult and costly process. Many researchers such as Self [2], have highlighted the intractable nature of this problem. Nevertheless, researchers recognize that although student models may not be highly accurate, and may not be complete from the cognitive perspective, they are indeed useful. In the traditional classroom approach, teachers also use less accurate student models; however, the teaching process is usually effective.

Perhaps the most commonly used strategy in student modeling is overlay modeling. When a learning system is constructed with overlay modeling, one of the first tasks to accomplish is to identify the concepts involved in the domain, and this process can be relatively easily performed by a human expert in the domain. However, there are other modeling techniques, such as perturbation models, which also incorporate incorrect knowledge. Even though the inclusion of student errors in their model provides some benefits, it also entails some problems. Specifically, the main problem of this modeling approach is the construction and maintenance of the bug library. The elicitation of this library is a time-consuming task which requires an exhaustive analysis of expert-student interactions and, accordingly, requires an expert with considerable experience. Despite this problem, the research on misconceptions has primarily focused on its diagnosis and remediation more than on its elicitation.

In this paper we present a technique for semi-automatic misconception discovery, primarily for declarative domains. The main goal is to provide the teachers with a collection of potential misconceptions. Subsequently, they have to decide whether or not they are misconceptions. A data-driven procedure is carried out based on the performance of students who have completed assessment activities (such as exercises, problems or test questions) in a certain subject domain.

The paper is structured as follows: The next three sections provide some background to the basis for this work. The first section contains a brief review of student modeling, focusing particularly on how researchers have approached the problem of modeling student error. Section 3 describes the state of the art in misconception modeling. Section 4 briefly looks at the basis of association rule algorithms, which have been used to develop the technique presented here. Then, the misconception inference technique is described in detail. Section 6 is devoted to the description of the experiment we have conducted to explore the performance of our technique. Finally, the conclusions we can extract from this work are outlined and also some of the research lines we plan to follow in the future.

2 Misconception Modeling

Holt et al. [3] reviewed the different types of student models. In their classification they identified the most commonly used *overlay models*, where the student is represented by his/her knowledge of a particular domain. Student behavior is compared to the expert behavior; therefore the student knowledge is a subset of the expert's. Differences between the two are assumed to be gaps in the student knowledge, but are not modeled specifically.

To overcome this problem, in the *perturbation models*, the student knowledge is *not only* a subset of the expert knowledge. In these kinds of models, the student may possess certain different knowledge in terms of both quantity and quality from that of the expert. Additionally, correct student knowledge (overlay modeling) is merged with the representation of faulty knowledge (misconceptions). This approach provides a more sophisticated model than other proposals. The set of misconceptions is usually stored in a *bug library* [4, 5].

The idea of bug libraries has considerable potential, but it also involves some problems [6]: The manual construction of the library is a difficult and time-consuming task; and the resulting library may not cover all the domain misconceptions, i.e., a student may have a misconception which is not included in the library.

3 Related Work

Most authors have focused on misconception diagnosis rather than on its elicitation, leaving this issue to the domain experts. Regarding the elicitation task, most of the research carried out in misconception modeling focuses on procedural errors, rather than declarative ones. The first efforts, in this area, were focused on automatically extending a bug library instead of creating it from scratch. Most notably, we can highlight the two rule-based algorithms INFER* and MALGEN [7], developed for the algebra equation domain and applied to high-school students. The first one creates new rules from incorrect actions taken by a student while solving a problem. MALGEN tries to elicit new rules automatically and without student intervention. Taking the domain rules as a starting point, it attempts to form new problem solving operators representing incorrect states by modifying existing operators. Both approaches required expert intervention to decide whether or not the inferred rule was suitable for the bug library.

ASSERT [6] is an algorithm for building tutoring systems using machine learning techniques to construct student models. It was the first system able to construct bug libraries automatically without needing the active participation of experts. To this end, it used a machine learning technique called *theory refinement* and has been applied to an introductory course of the C++ programming language in a rule-based tutoring system called NEITHER. The technique takes examples of student's behavior as input. If this behavior cannot be explained with the domain rules, the rules are modified and new ones are added to model the misconception.

MEDD [8] is an approach focused on the Prolog language programming domain. It is based on a similarity- and causality-based clustering technique of discrepancies between student programs and a set of reference programs. MEDD defines an error hierarchy containing error classes.

4 Itemsets and Association Rules

Nowadays most organizations have large databases with huge data sets. Nevertheless, because of the sheer amount of data available, important decisions are often taken using the intuition and experience of a human expert, instead of using the rich information stored [9]. In educational environments, the use of data mining techniques is growing [12]. We can find both techniques involving the analysis of the student-computer interaction logs and others studying other forms of educational data [13].

Association rules are an alternative means of extracting useful information from huge databases. They are used to establish the relationship (`if X=a then Y=b`) among different attribute values (i.e. if attribute `X` has the value `a`, the attribute `Y` will also have the value `b`). The algorithms which infer association rules take a set of transactions as input, each one labeled with a univocal identifier. Each transaction is formed by a set of attribute-value pairs which occur together.

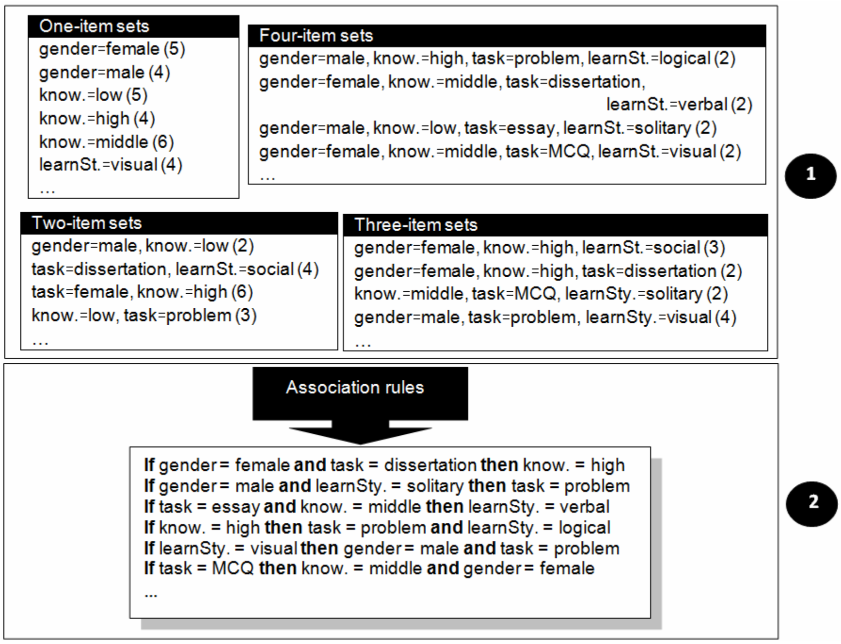

One-item sets
gender=female (5)
gender=male (4)
know.=low (5)
know.=high (4)
know.=middle (6)
learnSt.=visual (4)
...
Four-item sets
gender=male, know.=high, task=problem, learnSt.=logical (2)
gender=female, know.=middle, task=dissertation, learnSt.=verbal (2)
gender=male, know.=low, task=essay, learnSt.=solitary (2)
gender=female, know.=middle, task=MCQ, learnSt.=visual (2)
...
1
Two-item sets
gender=male, know.=low (2)
task=dissertation, learnSt.=social (4)
task=female, know.=high (6)
know.=low, task=problem (3)
...
Three-item sets
gender=female, know.=high, learnSt.=social (3)
gender=female, know.=high, task=dissertation (2)
know.=middle, task=MCQ, learnSty.=solitary (2)
gender=male, task=problem, learnSty.=visual (4)
...
Association rules
If gender = female and task = dissertation then know. = high
If gender = male and learnSty. = solitary then task = problem
If task = essay and know. = middle then learnSty. = verbal
If know. = high then task = problem and learnSty. = logical
If learnSty. = visual then gender = male and task = problem
If task = MCQ then know. = middle and gender = female
...
2

Fig. 1. Results of applying the Apriori method for mining the association rules of an educational dataset example

When inferring the association rules, the algorithms follow mainly two different phases:

1. The *itemset discovery*: In this stage, the most frequent collections of attribute-value pairs are found. Each value-attribute pair is called *item* and the set of items is called *itemset*. For this purpose, a minimum *support* must be defined a priori. The support is the frequency with which a set is found in the transaction dataset.
2. The *association rule inference*: Once the most frequent attribute sets are computed, this information is used as the basis of the association rules. Thus, rules are constructed taking the itemsets as antecedents or consequents. For each rule, its *support* is calculated, i.e. the fraction of transactions satisfying the rule, and its *confidence*, i.e. the number of data instances that the rule predicts correctly. A minimum support is indicated as an algorithm input parameter. At the end, those rules with a support lower than this threshold are discarded.

For instance, let us consider an educational domain example. Let us assume a dataset which contains information about the following four attributes, whose nominal values are included in brackets: `knowledge (low, middle, high)`, `learning style (visual, aural, verbal, physical, logical, social, solitary)`, `task (problem, MCQ, dissertation, essay)` and `gender (male, female)`. The goal is to mine association rules from these data. Figure 1 illustrates the results of applying the Apriori method [10], perhaps the most popular algorithm for mining association rules. This figure shows the results of the two phases of the algorithm. In the first (labeled number one), itemsets were discovered. Several itemsets were found and shown in different categories (see the boxes in the figure), in terms of the item number. Each item set also includes its *support* in brackets. In the example, the algorithm found that the itemset `knowledge=middle` appears six times and the set `gender=male, task=problem, learningStyle=visual` appears four times.

The lower part of Fig. 1 (labeled number two) shows the association rules inferred from the former itemsets. For example, the algorithm found that in the educational datasets, when the attribute `gender` is `male` and the `learning style` is `solitary`, then the `task` is `problem`.

5 A Technique for Misconception Inference

Our hypothesis is that if a student has a misconception, when he/she answers a question involving this misconception, he/she will tend to select those responses which best fit in with his/her incorrect knowledge state. Thus, these wrong responses could provide evidence about their misconception.

Let us consider, for instance, the algebra domain in which a student is solving questions in a test about fractions. If he/she does not know how to add fractions correctly, they may think, for example, that the fraction resulting from adding two fractions has a numerator equal to the sum of the numerators, and that the denominator is also the addition of the denominators. If in a test there are several questions involving the addition of two fractions, and these questions have an option where this addition is computed wrongly in the way that the student misunderstands, then he/she will select this response.

Consequently, if we have information available about student performance in a particular domain, we could try to discover the domain misconceptions by observing the selection patterns of wrong answers to questions, i.e., find out the most frequent associations among incorrect answers. The technique presented in this paper is based on applying an association rule inference algorithm to discover potential misconceptions.

Our technique takes as input the performances of a student population who took one or more tests about the domain whose misconceptions are being researched. Thus, we map the items which feed the association rule algorithms to the test questions, and the values of these items will be the options possible for answering a question. So the algorithm input is composed of student test sessions (equivalent to customer transactions). Each session contains a session identifier, the question posed and the set of choices selected for each question. To identify the misconceptions, we only pay attention to incorrect answers. For this reason, all the correct answers are filtered and blank responses (where allowed) are also discarded. Next, this “filtered” information is processed by the association rule algorithm. Consequently, it will return the most frequent sets of choices selected by the student sample and the frequency value, i.e. the number of times each choice association has been observed. In the post-processing, we order these sequences according to frequency. Our hypothesis is that the most frequent answer associations could correspond to misconceptions.

The next three subsections describe first the prior information needed by the technique and, later, the two stages of this technique which take place before and after the association rule algorithm execution.

5.1 The Evidence Model

The *Evidence-Centered Design* (ECD) proposal is a framework for designing, producing and delivering educational assessments [14]. ECD models incorporate representations of what a student knows and does not know, in terms of the results of his/her interaction performance (evidence) with assessment tasks [15]. In this line, we propose an evidence model which is based on the ECD framework. Our model (see Fig. 2) is composed of three layers: the *concept layer*, the *misconception layer* and the *task model*.

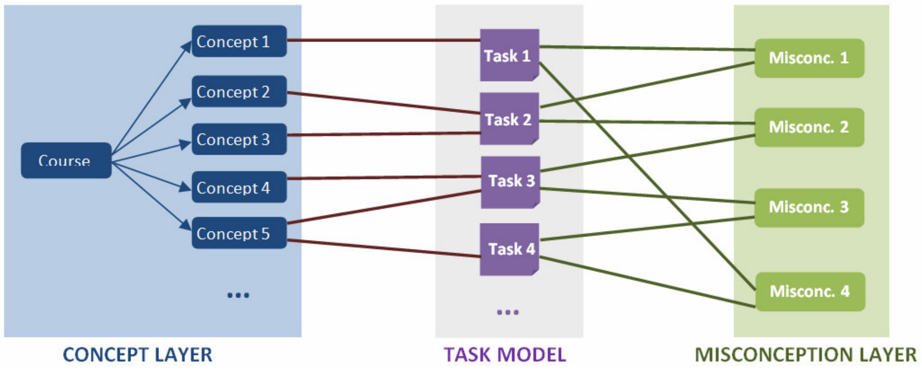


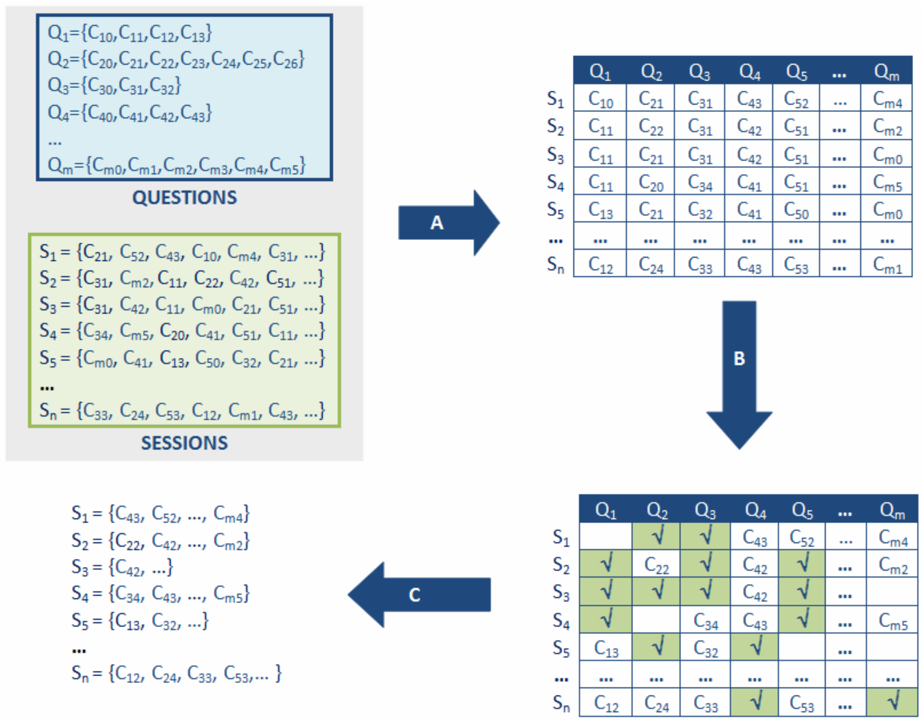
Fig. 2. The evidence model

In traditional teaching, the most popular tendency in the construction of domain models is to structure the contents of a course into parts, which are in turn divided into subparts, and so on. In this way, a hierarchy of variable granularity, called *curriculum* [16], is obtained. Curricula are often represented in intelligent learning environments by semantic networks, i.e., by directed acyclic graphs whose nodes are the pieces originated by the division of the course, and whose arcs represent relationships among the nodes. In the literature, a huge set of proposals exist (e.g., [17]) in which those parts have different names depending on their level in the hierarchy, e.g. topics, concepts, entities, chapters, sections, definitions, etc. Throughout this paper, all the nodes in the hierarchy will be generically called *concepts*. As Reye [18] states, concepts are curriculum elements which represent knowledge pieces or cognitive skills acquired (or not) by students.

From the point of view of student diagnosis, concepts are those elements susceptible to being assessed. However, we will not approach the structure and relationships among these concepts in this paper, since we consider it is not relevant for the work presented here. We only consider that all the concepts involved in the domain are collected in what we have called the *concept layer*.

In addition to this concept layer, our domain modeling approach is completed with a *misconception layer* which will be inferred semi-automatically. These misconceptions could be related to one (or even more) concepts of the former layer through the task model.

The *task model* is formed by assessment activities (e.g. exercises, problems or test questions), i.e., any kind of task which after been solved, could supply evidence about the student knowledge state. In this sense, we impose a restriction: the solutions to the activity reported by the student could be found in a finite number of states. That is, if a student takes a problem, his/her solution must reach a finite number of final states. To simplify, in this paper, we will consider that only one of these states is correct. Thus, the others supply evidence about the student's incorrect knowledge.



Q_1={C_{10},C_{11},C_{12},C_{13}}
Q_2={C_{20},C_{21},C_{22},C_{23},C_{24},C_{25},C_{26}}
Q_3={C_{30},C_{31},C_{32}}
Q_4={C_{40},C_{41},C_{42},C_{43}}
...
Q_m={C_{m0},C_{m1},C_{m2},C_{m3},C_{m4},C_{m5}}

QUESTIONS

S_1 = {C_{21}, C_{52}, C_{43}, C_{10}, C_{m4}, C_{31}, ...}
S_2 = {C_{31}, C_{m2}, C_{11}, C_{22}, C_{42}, C_{51}, ...}
S_3 = {C_{31}, C_{42}, C_{11}, C_{m0}, C_{21}, C_{51}, ...}
S_4 = {C_{34}, C_{m5}, C_{20}, C_{41}, C_{51}, C_{11}, ...}
S_5 = {C_{m0}, C_{41}, C_{13}, C_{50}, C_{32}, C_{21}, ...}
...
S_n = {C_{33}, C_{24}, C_{53}, C_{12}, C_{m1}, C_{43}, ...}

SESSIONS

	Q_1	Q_2	Q_3	Q_4	Q_5	...	Q_m
S_1	C_{10}	C_{21}	C_{31}	C_{43}	C_{52}	...	C_{m4}
S_2	C_{11}	C_{22}	C_{31}	C_{42}	C_{51}	...	C_{m2}
S_3	C_{11}	C_{21}	C_{31}	C_{42}	C_{51}	...	C_{m0}
S_4	C_{11}	C_{20}	C_{34}	C_{41}	C_{51}	...	C_{m5}
S_5	C_{13}	C_{21}	C_{32}	C_{41}	C_{50}	...	C_{m0}
...	...	...	...	...	...	...	...
S_n	C_{12}	C_{24}	C_{33}	C_{43}	C_{53}	...	C_{m1}

	Q_1	Q_2	Q_3	Q_4	Q_5	...	Q_m
S_1		√	√	C_{43}	C_{52}	...	C_{m4}
S_2	√	C_{22}	√	C_{42}	√	...	C_{m2}
S_3	√	√	√	C_{42}	√	...	
S_4	√		C_{34}	C_{43}	√	...	C_{m5}
S_5	C_{13}	√	C_{32}	√		...	
...	...	...	...	...	...	...	...
S_n	C_{12}	C_{24}	C_{33}	√	C_{53}	...	√

S_1 = {C_{43}, C_{52}, ..., C_{m4}}
S_2 = {C_{22}, C_{42}, ..., C_{m2}}
S_3 = {C_{42}, ...}
S_4 = {C_{34}, C_{43}, ..., C_{m5}}
S_5 = {C_{13}, C_{32}, ...}
...
S_n = {C_{12}, C_{24}, C_{33}, C_{53},... }

Fig. 3. Preprocessing stage of the Misconception Inference Technique

5.2 Preprocessing Stage

Before starting the inference procedure, a preprocessing stage is needed to prepare the input data. Fig. 3 illustrates graphically this procedure which takes as input the performances of students who took n assessment sessions: S_1, S_2, ..., S_n. Each student session could contain different exercises, all of them included in the following set: Q_1, Q_2, ..., Q_m. Exercises could be posed randomly to the students.

Let us assume, for instance, that these exercises are multiple-choice questions, where a question Q_i has r_i choices C_{i0}, C_{i1}, C_{i2}, C_{i3} and Cr_i. The first choice C_{i0} represents the blank response; the second, C_{i1}, the correct one; and the rest of them are incorrect answers. For example, in the session S_1 of Fig. 3, the student was posed question Q_1 which he/she left blank, Q_2 and Q_3 which were answered correctly, and finally, Q_4 and Q_m which were answered incorrectly.

The upper left side of Fig. 3 shows the set of questions and sessions before preprocessing. Firstly, we construct a matrix of student sessions versus the questions (transition A). Next, we identify and remove the correct question choices and the blank responses (transition B). Finally, data are prepared to be used by the association rule inference algorithm (transition C). As mentioned before, the input required by this algorithm is composed of transactions. In this case, each transaction represents the incorrect responses of a session. As a result, each transaction will be identified by the session identifier and will be composed of question-answer pairs. The lower left side of Fig. 1 shows the result of this preprocessing stage and hence the algorithm input.

5.3 Post-Processing Stage

The rule association algorithm provides the discovered sets of incorrect response associations (i.e. the itemsets), their support, and also the association rules. After that, the participation of the teachers (i.e. the domain experts) is required. For each itemset, the questions involved and the choices which led to this misconception are highlighted and subsequently, the teachers should reason on and discuss whether or not this itemset maps to a misconception. The itemsets should be presented according to their support, in reverse order. When the teachers identify an itemset which corresponds to a misconception, they should produce a description.

6 A Case Study

The experiment described in this section was carried out with undergraduate students of a Programming Fundamentals course, corresponding to the second semester of the first year of the B.Sc. in Telecommunications at the University of Málaga. The course curriculum is divided into five concepts:

1. *Files:* Main memory and secondary storage. Text and Binary files.
2. *Dynamic Memory:* Physical Management of Dynamic Memory. Pointers. Linked Lists: simple, double, circular.
3. *Recursion:* Concept. Physical Implementation. Use and Examples.
4. *Data Abstraction Introduction:* Data Abstraction and Object Oriented Programming. Basic concepts of Object Oriented Programming. Advanced concepts of Object Oriented Programming. Pointers to objects.
5. *Abstract Data Types (ADT):* Concept of ADT. Stack: Definition, Examples and Implementation. Queue: Definition, Examples and Implementation. Positional List: Definition, Examples and Implementation. Binary Trees: Definition, Examples and Implementation.

The course has around 300 students per academic year (average age, 18 years old) and a high academic failure rate. In 2009, we suggested that the teachers should use this approach to help them identify the most common misconceptions regarding student knowledge in this subject. Thus, the goal of the experiment was: first, to assist the teachers in the process of misconception discovery and, additionally, to explore the relationships among these misconceptions.

6.1 Study Design

To infer the Programming Fundamental course misconceptions, we needed information about the performances of students who had taken this course previously. Accordingly, we used the test and the data from an unpublished experiment we conducted in June of 2007. This experiment consisted of taking the same idea we applied to another course of Artificial Intelligence and Knowledge Engineering during the 2005-06 academic year [19]. We developed a new activity which was to construct a self-assessment test using our web-based adaptive system Siette. Consequently, the students had a drill-and-practice activity to prepare them for the final exam. The main goal of this test was to provide students with an environment to be able to train for the exam from their own homes. We call this type of test *open test*. In [19], empirical evidence suggested that these tests are useful for facilitating the student learning process.

The open test of this experiment had several restrictions: 1) each student was allowed to take the same test only once a day (this restrictive facility is provided by Siette and is configured during the test elicitation process). 2) Once the test was finished, the corrections were not shown and only the final score was supplied to the student. This restriction was included to force the students to try to complete the test, rather than simply copying the correct answers to the items, a strategy adopted by many students in other experiments. Instead of doing the actual test themselves, they wanted only to see the questions and the correct answers.

The test consisted of 20 multiple-choice questions, each one composed of a stem and a set of three choices. In each question, a blank response was allowed and only one choice was correct. All students always took the same set of questions but these were posed randomly each time.

6.2 Data Analysis and Results

A total of 233 sessions were collected from 103 individuals who participated in this experiment. Before any analysis, we computed Cronbach's alpha, which is commonly used as an estimator of the test result internal consistency. We obtained a value of 0.72, which is higher than the minimum validity threshold (0.70). Next we took the information from the students' performance and applied all the steps described in section 5: First, the performance data were expressed in vectors, identified by a session id, and formed by question-answer pairs. Then, we removed the pairs corresponding to correct answers and those which represented blank responses. After that, we applied the Apriori association rule inference algorithm with a minimum support of 25% and obtained the itemset collection and the association rules.

Regarding the itemsets (with more than one item), we only got itemsets of two items. More specifically, a total of 26 two-item sets were found. We sorted them in terms of their support, in reverse order. After that, we showed the results to the teachers graphically, i.e. the pair of questions of each itemset was displayed, and the corresponding answers included in the itemset were highlighted. Using this information, the teachers identified the seven misconceptions shown in Table 1. The first three columns contain the misconception description that the teachers produced after analyzing the itemsets, the itemset support and the concept which it is related to.

Table 1. Misconceptions identifies for the Programming Fundamentals domain

Misconception	*Support*	*Concept*	*% fail*
Misunderstanding of the use and management of the class method parameters	45	Data Abstraction Introduction	97%
Inability to seperate the implementation and the interface of a DAT	38	Abstract Data Types	89%
Misundestanding when differentiating between a Positional List (i.e. a DAT) and a linked list	36	Abstract Data Types	81%
Misundestanding of the mechanism of procedure calls in recursive algorithms	32	Recursion	68%
Misundestanding of the diference between binary and text files	31	Files	87%
Misunderstading of the way in which the elements of a Positional List can be modified	31	Abstract Data Types	81%
Misunderstanding of the assignment between objects of a class	28	Data Abstraction Introduction	89%

In addition, we computed the percentage of success and failure of those students who selected each itemset, i.e., for each itemset, we took the sessions of those students whose answers corresponded to the itemset. Then we calculated their test score and the percentage of sessions in which the score was less than 50%. As can be seen in the last column of Table 1, in all the cases but one, the percentage of students who failed was greater than 80%. This result suggests that, as well as considering the itemset support, the percentage of failure could help the teachers to determine the itemsets that actually correspond to misconceptions.

Concerning the association rules, we have not found any useful rule. Finally, we should mention that we conducted other executions of the association rule inference algorithm, reducing the threshold of the minimum support. As a consequence, we obtained more itemsets (a superset of the previous itemsets).No new misconceptions were discovered however.

7 Conclusions and Future Work

In this paper, we have presented a semi-automatic technique for discovering domain misconceptions. In general, this is a difficult task, usually carried out manually by human experts in the particular domain, who have observed over time the incorrect behavior of a large number of students. In this work, however, we have tried to automate the first part of the process normally done manually by experts. In the same way that association rule inference algorithms were initially developed to discover which products most consumers bought together, incorrect student answers can be correlated to student misconceptions. Using this premise, we have applied the association rule algorithms to discover potential misconceptions.

The experiment we have conducted, suggests that this technique can help teachers to identify the domain misconceptions. For this purpose, we use the itemsets discovered by the association rule algorithm. The experiment also suggests that data concerning the itemset support and the percentage of students who both selected this itemset and did not pass the assessment session, could be used to automatically discard those itemsets not corresponding to misconceptions.

In addition to the experiment described in this paper, we have carried out others with similar results. In all of them, a set of misconceptions is discovered from the itemsets but, regarding the association rules, either the algorithm did not identify any relevant misconceptions, or else it only expressed the association among answers.

Our technique helps teachers discover student misconceptions and, as a consequence, relate them with the tasks (questions or exercises) which provide evidence about this incorrect knowledge. In this sense, we are developing a well-founded and quantitative assessment model (as an extension of our previous work [20]) for updating student models which include misconceptions. As a result, after an assessment session, our model will supply the knowledge level estimation for each domain concept and also a "*misknowledge*" estimation for each domain misconception.

In addition, we are also working on an analogous technique for domain concept discovery, based on association rules. The goal is not only to discover the concepts, but the relationships among them.

Acknowledgments. This work has been co-financed by the Spanish Ministry of Science and Innovation (TIN2007-67515) and by the Andalusian Regional Ministry of Science, Innovation and Enterprise (P09-TIC-5105). Authors want to thank prof. Eva Millán for her comments on the final version of this paper.

References

1. Mayo, M., Mitrovic, A.: Optimising ITS Behaviour with Bayesian Networks and Decision Theory. International Journal of Artificial Intelligence in Education 12, 124–153 (2001)
2. Self, J.A.: Bypassing the intractable problem of student modeling. In: Frasson, C., Gauthier, G. (eds.) Intelligent Tutoring Systems: at the Crossroads of Artificial Intelligence and Education, pp. 107–123. Ablex, Norwood (1990)
3. Holt, P., Dubs, S., Jones, M., Greer, J.: The state of student modelling. In: Greer, E.J.E., McCalla, G. (eds.) Student modelling: The key to individualized knowledge-based instruction, vol. 125, pp. 3–35. Springer, New York (1994)
4. Brown, J.S., Burton, R.R.: Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science 2, 155–192 (1978)
5. Brown, J.S., VanLehn, K.: Repair theory: A generative theory of bugs in procedural skills. Cognitive Science 4, 379–426 (1980)

6. Baffes, P., Mooney, R.: Refinement-Based Student Modeling and Automated Bug Library Construction. International Journal of Artificial Intelligence in Education 7(1), 75–116 (1996)
7. Sleeman, D., Hirsh, H., Ellery, I., Kim, I.: Extending domain theories: two case studies in student modeling. Machine Learning 5, 11–37 (1990)
8. Sison, R., Numao, M., Shimura, M.: Multistrategy Discovery and Detection of Novice Programmer Errors. Machine Learning 38, 157–180 (2000)
9. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Academic Press, USA (2001)
10. Agrawal, R., Imielinski, T., Swami, A.: Mining Association Rules Between Sets of Items in Large Databases. In: SIGMOD Conference, pp. 207–216 (1993)
11. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
12. Romero, C., Ventura, S.: Educational Data Mining: A Survey from 1995 to 2005. Expert Systems with Applications 33(1), 135–146 (2007)
13. Baker, R.S.J.D., Yacef, K.: The State of Educational Data Mining in 2009: A Review and Future Visions. Journal of Educational Data Mining 1(1), 3–17 (2009)
14. Mislevy, R.J., Almond, R.G., Lukas, J.F.: A Brief Introduction to Evidence-Centered Design., CSE Report 632, National Center for Research on Evaluation, Standards and Student Testing (CRESST) (May 2004)
15. Shute, V.J., Graf, E.A., Hansen, E.: Designing adaptive, diagnostic math assessments for individuals with and without visual disabilities. In: PytlikZillig, L., Bruning, R., Bodvarsson, M. (eds.) Technology-based education: Bringing researchers and practitioners together, pp. 169–202 (2005)
16. Greer, J.E., McCalla, G.: Granularity-based reasoning and belief revision in student models. In: Greer, J.E., McCalla, G. (eds.) Student Modelling: The Key to Individualized Knowledge-Based Instruction, vol. 125, pp. 39–62. Springer, New York (1994)
17. Schank, R.C., Cleary, C.: Engines for Education. Lawrence Erlbaum Associates, Hillscale (1994)
18. Reye, J.: A belief net backbone for student modelling. In: Cerri, S.A., Gouardres, G., Paraguacu, F. (eds.) ITS 2002. LNCS, vol. 2363, pp. 596–604. Springer, New York (2002)
19. Guzmán, E., Conejo, R., Pérez-de-la-Cruz, J.L.: Improving Student Performance Using Self-Assessment Tests. IEEE Intelligent Systems 22(4), 46–52 (2007)
20. Guzmán, E., Conejo, R., Pérez-de-la-Cruz, J.L.: Adaptive Testing for Hierarchical Student Models. User Modeling and User-Adapted Interaction 17, 119–157 (2007)