

# Authoring Constraint-based Tutors in ASPIRE: a Case Study of a Capital Investment Tutor

Antonija Mitrovic<sup>1</sup>, Nicholas McGuigan<sup>2</sup>, Brent Martin<sup>1</sup>,  
Pramuditha Suraweera<sup>1</sup>, Nancy Milik<sup>1</sup> and Jay Holland<sup>1</sup>

<sup>1</sup>Intelligent Computer Tutoring Group  
University of Canterbury, Christchurch, New Zealand  
[anja.mitrovic@canterbury.ac.nz](mailto:anja.mitrovic@canterbury.ac.nz)  
[brent.martin@canterbury.ac.nz](mailto:brent.martin@canterbury.ac.nz)

<sup>2</sup>Lincoln University, Lincoln, New Zealand, [m McGuigan@lincoln.ac.nz](mailto:m McGuigan@lincoln.ac.nz)

**Abstract:** Although intelligent tutoring systems have proven their effectiveness, they are still not widely spread due to the high development costs. We present ASPIRE, a complete authoring and deployment environment for constraint-based intelligent tutoring systems. ASPIRE consists of the authoring server (ASPIRE-Author), which enables domain experts (i.e. teachers) to easily develop new ITSs for their courses, and a tutoring server (ASPIRE-Tutor), which deploys the developed systems. We describe the authoring process supported by ASPIRE, and illustrate it on the example of an ITS that teaches Capital Investment decision making to university students.

## Introduction

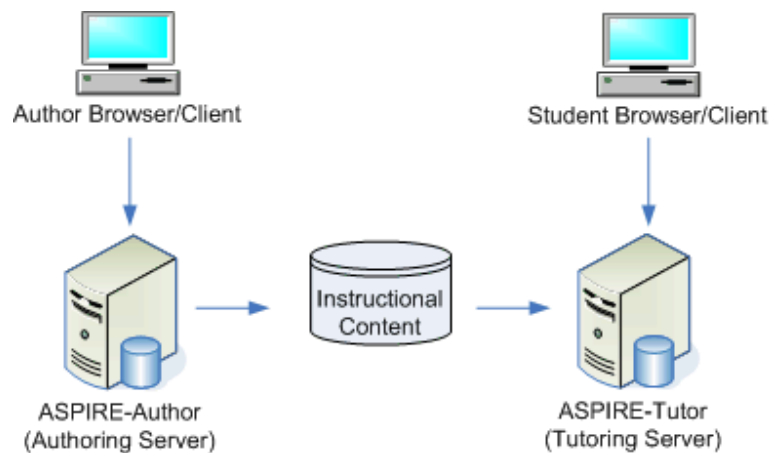
Intelligent Tutoring Systems (ITSs) have proven their effectiveness not only in controlled lab studies, but also in real classrooms (Koedinger et al., 1997; Mitrovic & Ohlsson, 1999; Mitrovic et al., 2004, 2007; VanLehn et al., 2005). These systems achieve significant improvements in comparison to classroom learning, due to the knowledge about the instructional domain, pedagogical strategies and student modelling capabilities. However, ITSs still have not achieved widespread effect on education due to their high complexity and difficulty of development. Composing the domain knowledge required for ITSs consumes the majority of the total development time (Murray, 2003). The task requires a multi-faceted expertise, including knowledge engineering, AI programming and the domain itself.

The Intelligent Computer Tutoring Group (ICTG) has developed a number of successful constraint-based tutors over the years in various domains, such as SQL queries (Mitrovic & Ohlsson, 1999; Mitrovic, 2003), database design (Suraweera & Mitrovic, 2004; Zakharov et al., 2005; Weerasinghe & Mitrovic, 2006), software analysis and design using UML (Baghaei & Mitrovic, 2007; Baghaei et al., 2007), English/vocabulary skills (Mayo & Mitrovic, 2001; Martin & Mitrovic, 2003), and also for procedural skills, such as data normalization within relational database design (Mitrovic, 2005) and logical database design (Milik et al., 2006). Although constraint-based tutors are easier to develop in comparison to some other existing types of ITSs (Mitrovic et al., 2003), their development is still a labour-intensive process that requires expertise in Constraint-Based Modelling (Ohlsson, 1992; Ohlsson & Mitrovic, 2007) and programming. In order to reduce the time and effort required for producing constraint-based tutors, we developed ASPIRE, an authoring system that can generate the domain model with the assistance of a domain expert and produce a fully functional system.

We start with a brief introduction to ASPIRE, including an outline of the authoring process and the architecture of the system. Section 3 presents CIT, the Capital Investment Tutor, focusing on the authoring process, while the following section presents the results of an evaluation study. Finally, Section 5 presents conclusions and the directions of future work.

## ASPIRE

ASPIRE consists of an authoring server (ASPIRE-Author) which assists the author in developing new ITSs, and a tutoring server (ASPIRE-Tutor) that delivers the resulting ITSs to students (see Figure 1). ASPIRE-Author makes it possible for the human expert (the author) to describe the instructional domain and the tasks the students will be performing, as well as to specify problems and their solutions. Both servers are implemented in Allegro Common Lisp as web servers for users to interact through a standard web browser. All required domain-dependent information, such as the domain model and other configuration details produced by ASPIRE-Author, are transferred to ASPIRE-Tutor in the XML format. We do not present details of ASPIRE in this paper; the interested reader is referred to (Mitrovic et al., 2006) and the ASPIRE Web page (<http://aspire.canterbury.ac.nz>).



**Figure 1.** The Architecture of ASPIRE

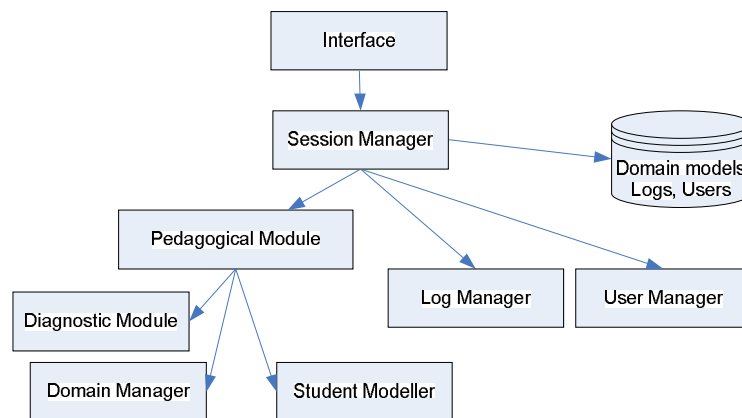
The authoring process in ASPIRE-Author consists of eight steps shown in Figure 2. Initially, the author creates a new domain and specifies its general features, such as whether the domain consists of a sub-domains focusing on specific areas, and whether the domain is procedural or not. In the case of procedural domains, the author is required to enumerate the problem-solving steps. The author then describes the domain in terms of an ontology (step 2), which identifies the important domain concepts, their properties and mutual relationships. In step 3 the author specifies the structure of problems and solutions. The author is then asked to design the student interface (step 4) and provide a set of problems and their solutions (step 5). Syntax constraints are generated automatically (step 6) by analysing relationships between concepts and properties of concepts specified in the ontology. The syntax constraint generation algorithm extracts the restrictions specified for relationships and properties and translates them into constraints. These constraints are applicable to both procedural and non-procedural domains. An extra set of constraints are generated for procedural domains to ensure that the student adheres to the correct problem-solving procedure. In step 7, ASPIRE generates a set of constraints for checking the semantics of the answer using a machine-learning technique: alternative (correct) solutions are compared, and, if necessary, constraints are specialized or generalized to be consistent across all the given solutions. Semantic constraints enable the ITS to model alternative correct solution approaches. Finally, the author deploys the system in step 8. As the focus of this paper is on CIT, a system developed in ASPIRE, we do not present the details of the constraint generation algorithms; the interested reader is referred to (Suraweera et al., 2005).

- |   |
|---|
| <ol style="list-style-type: none"><li>1. Specifying the domain characteristics</li><li>2. Composing the domain ontology</li><li>3. Modelling the problem and solution structures</li><li>4. Designing the student interface</li><li>5. Adding problems and solutions</li><li>6. Generating syntax constraints</li><li>7. Generating semantic constraints</li><li>8. Deploying the tutoring system</li></ol> |
|---|

**Figure 2.** The steps of the authoring process

The architecture of ASPIRE-Tutor is illustrated in Figure 3. The interface module is responsible for producing an interface for each deployed ITS. The interface provides features such as login/logout, select/change domains/problems, submit solution for evaluation etc. The session manager is responsible for maintaining the state of each student during their interaction. The current session state is described by information such as the selected domain, sub-domain and problem number. The session manager also acts as the main entry point to the system, invoking the relevant modules when needed. For example, when a student submits a solution to be validated, the session manager passes on all information to the pedagogical module, which returns the feedback to be presented to the student. The Pedagogical Module (PM) decides how to respond to each student request. It is responsible for handling all pedagogy-related requests including selecting a new problem, evaluating a student's submission and viewing the student model. In the event of evaluating a student's submission and providing feedback, the PM delegates the task of evaluating the solution to the diagnostic module and decides on the appropriate feedback by consulting the student model. The student modeler maintains a long-term model of the student's knowledge.

ASPIRE-Tutor serves the developed domain models as web-based tutors. In addition to running a collection of tutoring systems in parallel, ASPIRE-Tutor also provides the functionality for managing user accounts (for administrators, teachers, students and authors). It also provides functionality for teachers to tailor an ITS to his/her specific needs for a particular class, and allows teachers to assign ITSs to their students.



**Figure 3.** The architecture of ASPIRE-Tutor

ASPIRE has been evaluated in several domains, ranging from fraction addition to thermodynamics and engineering mechanics. In this paper, we present the Capital Investment Decision; see (Suraweera et al., 2007) for details of the evaluations in other instructional domains.

## CIT - The Capital Investment Tutor

Capital investment decision making plays a crucial role in the financial evaluation of non-current assets within contemporary organisational practice. Our teaching experience shows that capital investment evaluation techniques, namely the *accounting rate of return*, *net present value* and the *internal rate of return*, are problematic for students to master. Students find the principles of capital investment decision making difficult to comprehend, with a lack of ability to translate from theory to practice. It was envisaged that the CIT would enable students to apply theoretical financial decision making to 'real-life' simulated business environments.

It was with this in mind that the CIT was developed by the second author of this paper as one of the crucial evaluation stages of the ASPIRE project. In this section we describe the process of developing CIT.

Figure 4 shows a screenshot of the *Domain* tab of ASPIRE-Author, which corresponds to the first step of the authoring procedure, in which the author describes the domain and specifies problem-solving steps. The task the student needs to perform is a procedural one, consisting of seven steps. In the first step, the student constructs a timeline of project costs from the information given in the problem statement. This step will be shown to the student on its one, on the first page. In step two (shown on its own on the second page), the student needs to identify the relevant problem type, in terms of the variable which needs to be calculated. Step 3 requires the student to select the

formula corresponding to the chosen variable, and then enter the parameters for the formula in step 4. In step 5 the student enters the known values into the selected formula, and then specifies the computed value in step 6. Based on the computed value, the student then makes the final decision regarding capital investment in step 7. In CIT, there is only a single problem set, although ASPIRE allows for multiple problem sets to be defined

The screenshot shows the 'Domain Details' section of the ASPIRE Authoring System. The domain name is 'Capital Investment Decision' and the description is 'A computer aided tutorial programme to assist students with capital investment decision making'. The type is set to 'Procedural'. Below this, a table lists seven procedural steps with checkboxes for 'New Page', 'Page no', 'Problem Specific Instruction', and 'Repeatable'. The 'Problem Sets' section below the table shows one set named 'set1' with the description 'Problem set 1'. A 'Save structure' button is visible at the bottom of the domain details section. The footer includes 'Logout ASPIRE-Tutor' and 'Powered by ASPIRE'.

Procedural Steps						
	Name	Task Description	New Page	Page no	Problem Specific Instruction	Repeatable
<input type="checkbox"/>	1 Construct a Timelin	Display visually on a tim	<input checked="" type="checkbox"/>	1	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	2 Problem Type	Choose relevant type of	<input checked="" type="checkbox"/>	2	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	3 Selecting the formu	Select the formula that c	<input checked="" type="checkbox"/>	3	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	4 Specifying the valu	Specify the values of n &	<input type="checkbox"/>	3	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	5 Completing the exp	Enter the missing value	<input type="checkbox"/>	3	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	6 Calculating the NPV	Calculate the value of N	<input type="checkbox"/>	3	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	7 Making the decisio	Make the decision	<input checked="" type="checkbox"/>	4	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Problem Sets		
	Name	Description
<input type="checkbox"/>	1 set1	Problem set 1

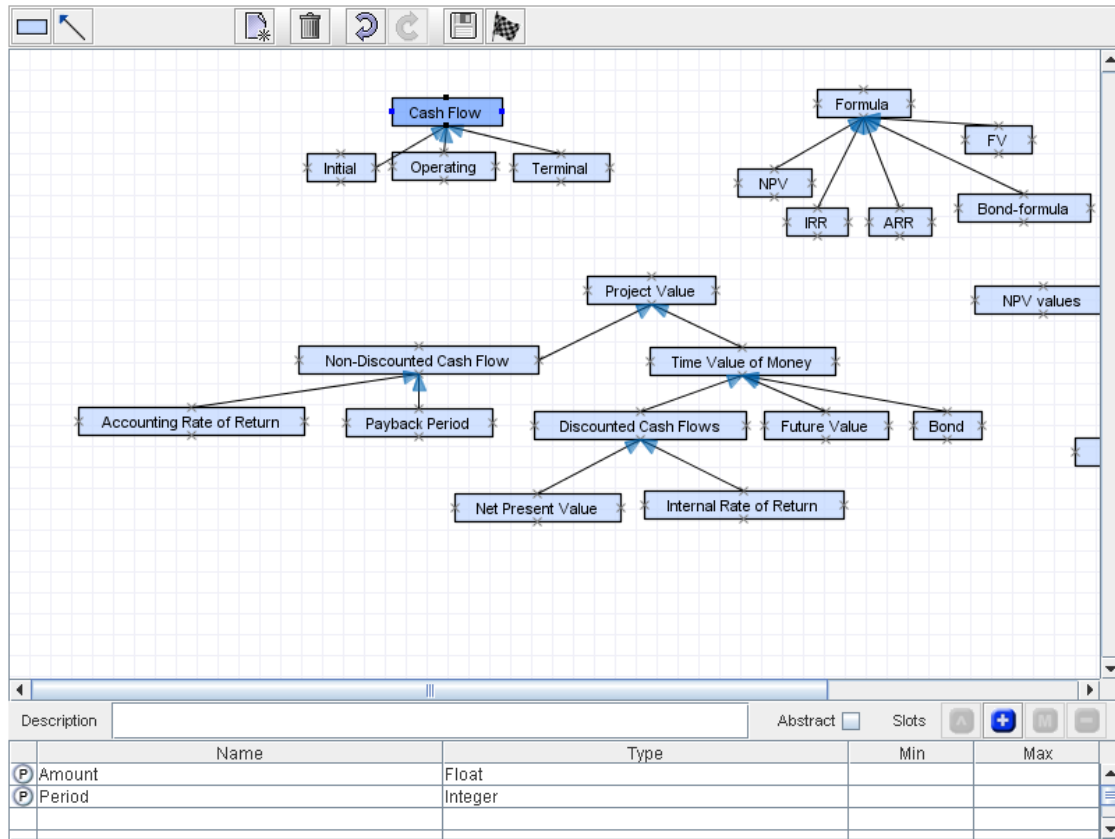
**Figure 4.** The steps of the Capital Investment Decision domain

The author then specifies the ontology for this instructional domain. Composing a domain ontology is, in general, a much easier task for authors than composing constraints manually (Suraweera et al., 2004). ASPIRE-Author provides an ontology workspace for visually modelling ontologies (Figure 5). The ontology describes the domain by identifying important concepts and relationships between them. The ontology outlines the hierarchical structure of the domain in terms of sub- and super-concepts; for example, the *Cash Flow* concept in Figure 5 is specialised into *initial*, *operating* and *terminal* cash flows. Each subconcept inherits the properties of its parent concept, but also might have its own properties, and may be related to other domain concepts. The specialization relationships between domain concepts are visually represented as arrows between concepts. The *Cash Flow* concept is the currently selected concept in Figure 5, and its properties (*Amount* and *Period*) are shown in the table below.

In the third authoring step, the author specifies the problem/solution structures. Problems can consist of components (textual or graphical) and a problem statement. In our domain, each problem has a problem statement and an attached photograph. For each step of the procedure, the student's solution might contain one or more components. The overall structure of solutions depends on whether the domain is procedural or not. The solution structure for capital investment decision making is outlined in Figure 6.

The student interface needs to be designed next. The final outcome of this phase is an interface in which students compose their solutions to problems. ASPIRE generates a default interface, which is form-based, by placing an input area for each component defined in the solution structure. The author can replace one or more pages with Java applets, which might be more suited to the task the student is performing than a textual interface. We developed two applets for CIT, which will be discussed later. After designing the student interface, the author entered twelve problems and their solutions. In this domain, there is only one correct solution per problem; in other domains, there might be multiple correct solutions, and in such cases the author would need to specify all valid solutions, which are used by the authoring system to generate semantic constraints.

**Domain Ontology (Domain - Capital Investment Decision)**



**Figure 5.** Ontology for the Capital Investment domain

Once example problems and their solutions are available, ASPIRE-Author generates the domain model. In case of CIT, there are 34 syntax and 25 semantic constraints generated. Syntax constraints check whether the student’s solution follows the syntactic rules of the domain; they make sure that all the necessary solution components are specified, that they are of appropriate types, and are related to other solutions components as necessary. Figure 7 illustrates one syntax constraint, which checks whether the student has specified one part of the solution (operational cash flows). The author cannot change the constraint itself, but can modify the feedback messages attached to it, to tailor them in order to be more understandable to students in comparison to the automatically generated feedback messages.

Problem solving step	Solution components
1. Construct a timeline	Cash flows (initial, operating and terminal)
2. Identify problem type	One of <i>Accounting Rate of Return</i> , <i>Bond</i> , <i>Future Value</i> , <i>Internal Rate of Return</i> , <i>Net Present Value (NPV)</i> , <i>Payback Period</i>
3. Select the formula	One of the pre-specified set of formulae
4. Specify the parameters for the formula	$n, k$
5. Complete the formula	All components of the NPV formula
6. Enter the NPV value	NPV value
7. Make the final decision	Decision

**Figure 6.** Solution structure for CIT

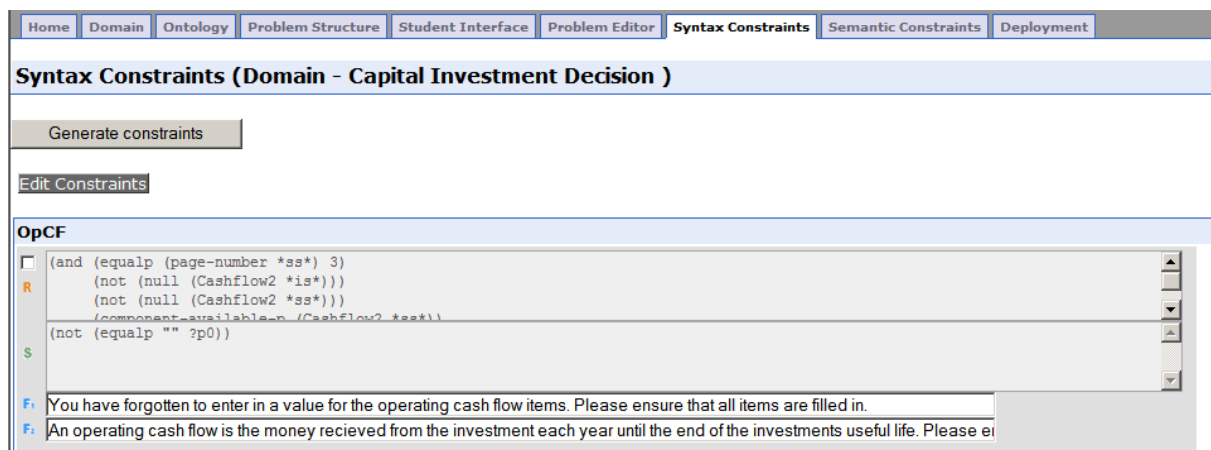


Figure 7. A syntax constraint from CIT

Finally, the author deploys the domain, which results in the domain information being transferred to ASPIRE-Author. The ITS is immediately available for the author to try out. The author/teacher can also define one or more groups, consisting of students who will have access to the system. The teacher can tailor the behaviour of the system to his/her needs. For example, the teacher can specify the feedback levels to be offered, as well as the progression between them. ASPIRE offers the following feedback levels: *Quick Check* (specifying whether the answer is correct or not), *Error Flag* (identifying only the part of the solution that is erroneous), *Hint* (identifying the first error and providing information about the domain principle that is violated by the student's solution), *Detailed Hint* (a more detailed version of the hint), *All Errors* (hints about all errors) and *Show Solution*. By default, ASPIRE starts with *Quick Check* and progresses with each consecutive submission to the same problem to *Detailed Hint*, unless the student asks for a specific type of feedback. Information about all errors and the solution are only available at request. However, the teacher can override this default behaviour by limiting the types of feedback, prohibiting the full solution from being shown, specifying the minimal number of attempts before the full solution can be seen, or by specifying the maximal level of feedback to be provided automatically. The teacher can also specify the problem selection mechanisms available to students.

Figure 8 shows the student interface with the applet for the first step in CIT. The top area of the page provides controls for selecting problems, obtaining help, and changing/leaving the ITS. The problem statement is shown together with the photo describing the situation. The problem-solving area for this step consists of an applet, visualizing the time line. The student needs to label the periods on this timeline, and enter the amounts corresponding to the various types of cash flows. In the situation illustrated in Figure 8, the student has incorrectly labelled the initial time as period 1 on the timeline, entered the incorrect value for the initial cash flow, and has not specified the rest of the timeline. The feedback shown in the right-hand panel corresponds to the *All Errors* level: the first hint discusses operating cash flows that are missing, the second one discusses the initial cash flow for which the student supplied the wrong value (3,5000 instead of 10,000 as specified in the problem text), while the last one discusses the terminal cash flow. The first and third hints come from violated syntax constraints, while the second one comes from a violated semantic constraint. The student can change the solution based on the feedback provided, and submit the solution to CIT again.

Since CIT is a procedural tutor, the student needs to complete each step correctly before being allowed to move on to the following step. Figure 9 shows a situation when the student has already completed the first four steps successfully: the student has completed the timeline, selected NPV as the appropriate evaluation mechanism to be used in the problem, selected the correct formula for NPV, and specified the correct values for  $n$  (the number of years) and  $k$  (the interest rate), which are the parameters used in the formula for NPV. The applet shows the current step, in which the student is to fill in the values into the formula for calculating NPV. Please note that the applet shows the expanded version of the summation formula, showing the four terms corresponding to the operating cash flows and the last term ( $pi$ ) corresponding to the initial cash flow. The goal of this step is to check whether the student can differentiate between the initial cash flow (the term subtracted from the others) from the operating cash flows, and also whether the student understands the various time periods involved and corresponding cash flows. The student has specified the initial cash flow correctly, but has not specified any of the operating cash flows and the

corresponding time periods. The feedback shown in Figure 9 is on the *Hint* level, and discusses the operating cash flows which are missing from the student's solution.

**Figure 8.** Student interface showing the first step of the procedure (the timeline)

We have chosen to develop applets for these steps to make CIT visually more attractive, although the task itself can be executed using purely textual input. However, in other domains it might be impossible to use a textual interface. For example, if the student needs to develop some kind of diagram, a drawing applet would be necessary. Our experience shows that the time and effort needed to develop applets is higher than the requirements for the other development tasks in ASPIRE. Additionally, applets require software engineering experience, and for that reason we do not expect teachers to be able to develop applets by themselves.


## Evaluation Study

We trialled CIT in May 2007 in ACCT102, Accounting and Finance for Business, an introductory business decision making course taught at the Lincoln university. There were 32 students who volunteered to participate in this study. The students have listened to lectures covering the relevant material before using the system. The students used CIT during scheduled tutorials of 50 minutes. During this time, the students sat a short pre-test, interacted with the system, and then sat a post-test and filled in a user questionnaire. The average interaction time with CIT was only 30 minutes. The mean results on the pre-test was 25% (sd=14%), while the students achieved significantly higher results (p=0.07) on the post-test with the mean of 35% (sd=31%). We attribute the relatively low results on the post-test to the short session length.

**Capital Investment Decision**

Select one... Next Problem System's Choice Help Change ITS Exit ITS

---

**Problem**  
 Calcady Engineering Ltd is considering a capital expenditure that requires an initial investment of \$10 000 and provides a net cash flow of \$3 500 per year for four years. The firm has a required return of 12 percent.  
 photo 

**Solution workspace**

$$\sum_{t=1}^n \frac{CF_t}{(1+k)^t} - \pi$$

Supply Additional Elements

n= 4      k= 0.12

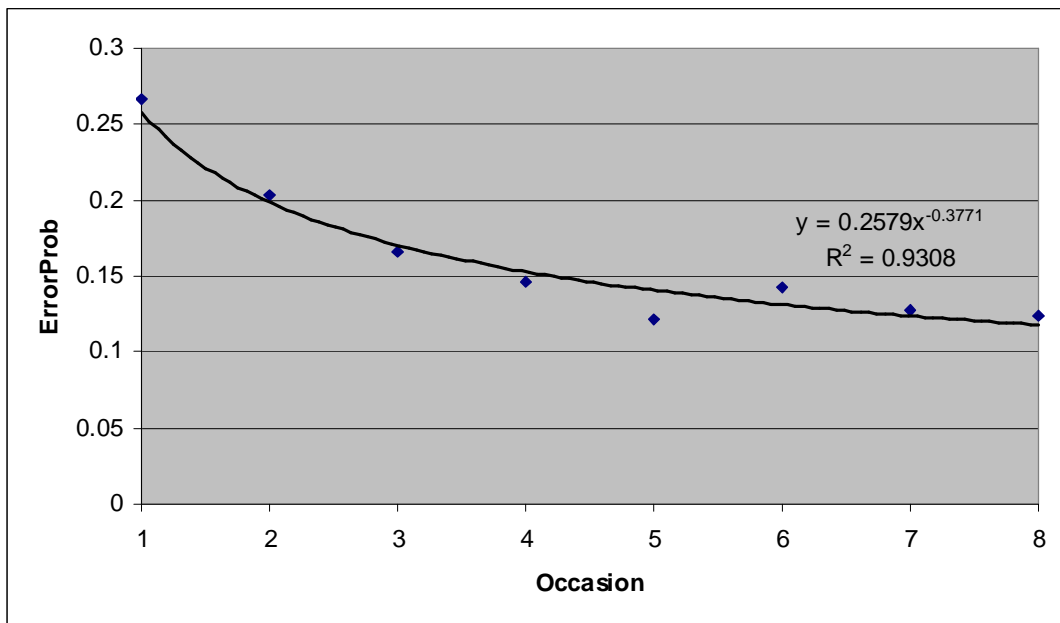
Fill in the Expanded Equation

CF1	+	CF2	+	CF3	+	CF4	-	1000
$(1 + 0.12)^{Y1}$		$(1 + 0.12)^{Y2}$		$(1 + 0.12)^{Y3}$		$(1 + 0.12)^{Y4}$		

Logout

Powered by **ASPRE**

**Figure 9.** Student interface showing the fifth step of the problem



**Figure 10.** The learning curve from the CIT study



We have performed a preliminary analysis of the data collected during the study. Figure 10 shows the learning curve. The x-axis shows the opportunity to use a specific knowledge element (i.e. constraint) during the session. The y-axis shows the probability of making errors on the same knowledge elements. The data points were averaged over all constraints and all students who interacted with CIT. The data points have an excellent fit to the power curve, showing that students do learn effectively in CIT. Furthermore, the learning rate (i.e. the exponent of the power curve equation) is very high, showing that students acquire the necessary knowledge fast. The initial error probability of 0.27 dropped by more than 50% to 0.12 after only eight attempts on average, which is a significant result, especially taking into account that the session length was short. The students' comments from the questionnaire show that the students enjoyed interacting with the system and believed that their understanding of the domain was improved as a consequence of using CIT. The students pointed out the usefulness of feedback in correcting their mistakes. We plan to repeat the evaluation study in early 2008, with longer sessions (2 hours). In the next study, we will compare the learning performance of students using CIT to that of students learning in a traditional way, in a group situation with a human tutor.

## Conclusions

We provided an overview of ASPIRE, an authoring system that assists domain experts in building constraint-based ITSs and serves the developed tutoring systems over the web. ASPIRE follows a semi-automated process for generating domain models, in which the author is required to provide a description of the domain in terms of an ontology, specify the structure of problems and solutions, and provide examples of both. From this information, ASPIRE induces the domain model, and produces a fully functional web-based ITS, which can then be used by students. ASPIRE also provides additional support for administrators to create user accounts and maintain the activities in ASPIRE. It also supports teachers in tailoring ITSs to their classes. The paper presented the process of developing CIT, an ITS that teaches Capital Investment decision making, in ASPIRE. We discussed the phases of CIT's development, as well as the result of a preliminary evaluation, which showed that the students do learn from interacting with CIT. We plan to conduct a longer study of CIT effectiveness.

CIT is only one of the ITSs developed in ASPIRE. We have also been developing ITSs in ASPIRE for the areas of thermodynamics, engineering mechanics, chemistry and arithmetic. The authors involved in this work have various types of backgrounds, ranging from teachers to Computer Science postgraduate students. Our experiences show that although initially authors need to learn about ontologies and the development philosophy supported by ASPIRE, they find it a flexible and powerful tool. ASPIRE is freely available on the Web, and we do hope that other teachers will be using it to develop ITSs for their courses.

## Acknowledgements

The ASPIRE project was supported by the eCDF grants 502 and 592 from the Tertiary Education Commission of New Zealand. We thank all members of ICTG for their support.

## References

- Baghaei, N., Mitrovic, A. & Irwin, W. (2007) Supporting Collaborative Learning and Problem-Solving in a Constraint-based CSCL Environment for UML Class Diagrams. *Computer-Supported Collaborative Learning*, 2(2-3), 159-190.
- Baghaei, N., Mitrovic, A. (2007) From Modelling Domain knowledge to Metacognitive Skills: Extending a Constraint-based Tutoring System to Support Collaboration. *User Modeling 2007*, 217-227.
- Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A., (1997) Intelligent Tutoring goes to School in the Big City. *Artificial Intelligence in Education*, 8, 30-43.
- Mayo, M. & Mitrovic, A. (2001) Optimising ITS Behaviour with Bayesian Networks and Decision Theory'. *Artificial Intelligence in Education*, 12(2), 124-153.
- Martin, B. & Mitrovic, A. (2003) Domain Modeling: Art or Science? *Artificial Intelligence in Education 2003*, IOS Press, 183-190.

- Milik, N., Marshall, M., Mitrovic, A. (2006) Responding to Free-form Student Questions in ERM-Tutor. *Intelligent Tutoring Systems 2006*, 707-709.
- Mitrovic, A. (2003) An Intelligent SQL Tutor on the Web. *Int. J. Artificial Intelligence in Education*, 13(2-4), 173-197.
- Mitrovic, A. (2005) The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor. *Artificial Intelligence in Education 2005*, IOS Press, 499-506.
- Mitrovic, A. & Ohlsson, S. (1999) Evaluation of a Constraint-Based Tutor for a Database Language. *Artificial Intelligence in Education*, 10(3-4), 238-256.
- Mitrovic, A., Koedinger, K. & Martin, B. (2003) A Comparative Analysis of Cognitive Tutoring and Constraint-Based Modelling. *User Modeling 2003*, Springer-Verlag, pp. 313-322.
- Mitrovic, A., Suraweera, P., Martin, B. & Weerasinghe, A. (2004) DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. *Journal of Interactive Learning Research*, 15, 409-432.
- Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N. & Holland, J. (2006) Authoring Constraint-based Tutors in ASPIRE. *Intelligent Tutoring Systems 2006*, 41-50.
- Mitrovic, A., Martin, B. & Suraweera, P. (2007) Intelligent Tutors for all: Constraint-based Modeling Methodology, Systems and Authoring. *IEEE Intelligent Systems*, 22(4), 38-45.
- Murray, T. (2003) An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art. *Authoring tools for advanced technology learning environments*. 491-545.
- Ohlsson, S. (1992) Constraint-based student modeling. *Artificial Intelligence and Education*, 3(4), 429-447.
- Ohlsson, S. & Mitrovic, A. (2007) Fidelity and Efficiency of Knowledge Representations for Intelligent Tutoring Systems. *Technology, Instruction, Cognition and Learning*, 5(2), 101-132.
- Suraweera, P. & Mitrovic, A., (2004) An Intelligent Tutoring System for Entity Relationship Modelling. *Artificial Intelligence in Education*, 14, 375-417.
- Suraweera, P., Mitrovic, A. & Martin, B. (2004) The Role of Domain Ontology in Knowledge Acquisition for ITSs. *Intelligent Tutoring Systems ITS 2004*, Springer-Verlag, 207-216.
- Suraweera, P., Mitrovic, A. & Martin, B. (2005) A Knowledge Acquisition System for Constraint-based Intelligent Tutoring Systems. *Artificial Intelligence in Education*, IOS Press, 638-645.
- Suraweera, P., Mitrovic, A., Martin, B. (2007) Constraint Authoring System: an empirical evaluation. *Proc. 13<sup>th</sup> Int. Conf. Artificial Intelligence in Education AIED 2007*, 451-458.
- VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A. & Wintersgill, M. (2005). The Andes Physics Tutoring System: Lessons Learned. *Artificial Intelligence in Education*, 15, 147-204.
- Weerasinghe, A., Mitrovic, A. (2006) Facilitating Deep Learning through Self-Explanation in an Open-ended Domain. *Knowledge-based and Intelligent Engineering Systems*, IOS Press, 10(1), 3-19.
- Zakharov, K., Mitrovic, A., Ohlsson, S. (2005) Feedback Micro-engineering in EER-Tutor. *Artificial Intelligence in Education 2005*, IOS Press, 718-725.