An Approach of Implementing General Learning Companions for Problem Solving

Chih-Yueh Chou, Tak-Wai Chan, and Chi-Jen Lin

Abstract—Adding a learning companion, a computer simulated social agent, to a computer based learning system can enhance its educational value by enriching the way in which the computer and the user interact. This paper presents a novel simulation approach, named General Companion Modeling (referred to hereinafter as GCM), to implement learning companions in a general problem-solving domain. DwestAgent, a learning companion system, is also implemented using the GCM approach to demonstrate the feasibility of the proposed approach. In addition, the approach can help developers of learning companions clarify implementation issues and requirements involved in simulating 1) domain competencies, 2) learning competencies, 3) behaviors as a peer tutor, and 4) behaviors as a peer tutee of a learning companion. Using GCM, one can simulate learning companions with various characteristics by adjusting parameters within the proposed simulation framework.

Index Terms—Learning companion, educational agent, social learning, collaborative learning, computer assist learning, problem solving, simulation.

1 Introduction

Computer-based learning systems provide an attractive environment for learning skills of problem solving. Students can do problem-solving exercises as well as access their learning data. With proper designs, these learning systems can assist students in forming learning strategies and structuring their knowledge. In addition, adding a computer-simulated tutor to guide a student's learning can enhance the educational value of a learning environment. For example, Burton and Brown [1] developed a computer board game system, named West, with a coach. West was designed for practice arithmetic skills. That investigation also proposed several principles to guide tutoring and discussed how to implement the adopted philosophy, diagnostic modeling, and tutorial strategies of the system. West is a typical intelligent tutoring system (ITS) that contains a computer tutor to observe and guide the student's movements.

Several researchers have conferred on the role of the computer as a collaborative partner of the user, not only an authorized teacher [2], [3], [4], [5]. Chan et al. developed a distributed learning companion system (LCS), named Distributed West [6], which is a reimplementation of West. Distributed West, a distributed system, consists of two connected computers so that students can learn collaboratively and/or competitively at different locations. In addition, a student can interact with a computer companion and/or a computer teacher in a centralized environment. Distributed West contains 768 possible learning models based on different combinations of agents and dimensions of factors, such as role of the computer teacher, role of the learning companion, learning format, relationship, and level of the agents. The computer teacher may function as a tutor, coach, critic, or evaluator. An evaluator is nonadaptive while a tutor, coach, or critic can be either adaptive (sensitive to the student's performance history) or nonadaptive.

As a computer-simulated student, the learning companion can compete against or collaborate with human students [4], [7]. The learning companion can function as a competitor, tutee, or tutor to provide users with a social learning environment. The learning companion can be used to encourage user reflection and articulation [8], increase the user's motivation [9], support learning activities of "reciprocal tutoring" [10], [11], or support strategy of "learning by disturbing" [12] or "learning by teaching" [13], [14], [15], [16]. Additionally, Scott and Reif indicated that reciprocal tutoring with a learning companion is nearly as effective as individual tutoring by human expert teachers and more effective than a well-taught class [11].

Among the 768 possible models of Distributed West, three models are implemented and evaluated in this study. In Model one, two students collaborate with each other in competing against a computer opponent. In Model two, a student competes against another one under the supervision of a computer teacher. Model three resembles Model two, except no computer teacher is involved. In the three models, the learning companion is implemented only as a competitor, not as a collaborator. This study implements the learning companion not only as a competitor, but also as a collaborator. In doing so, two collaborative learning models are proposed: 1) the user functions as a peer tutor, helping a learning companion to compete against another learning companion, and 2) the user functions as a peer tutee, which is under the supervision of a learning companion to compete against another learning companion. Either a learning companion or a human student, a peer tutor, guides a peer tutee. The peer tutee can also be a learning

• *The authors are with the Institute of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan, R.O.C. E-mail: {yueh, chan, zen}@src.ncu.edu.tw.*

Manuscript received 3 May 1999; revised 28 Feb. 2001; accepted 2 Mar. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 109743.

companion or a human student. A peer tutor functions exactly like a tutor, except that the peer tutor is not an expert and may make erroneous recommendations. Therefore, the peer tutee must determine whether the peer tutor's recommendation is correct.

Using a learning companion, not a human, to compete or collaborate with a user has two advantages. First, the system can control the competency and behavior of the learning companion for a particular pedagogical strategy. Hietala and Niemirepo suggested that a group of learning companions with heterogeneous domain competency placed at a user's disposal increases the user's motivation [9]. Goodman et al. designed a learning companion, Lucy, whose behavior is controlled for encouraging user reflection and articulation [8]. Second, the system is available to the user to determine the learning companion's competency and behavior. Hietala and Niemirepo demonstrated that different users prefer different learning companions [9]. In their experiments, the system provides two strong learning companions with an expert's competency and two weak learning companions with a nonexpert's competency. The extraverts and subjects with a lower IQ preferred asking the weak companions, while the introverts and subjects with a higher IQ preferred asking the strong companions.

However, the implementation issues of a learning companion for different learning activities and pedagogical strategies have seldom been addressed. This study addresses the following implementation requirements and related issues to simulate learning companions. An approach, named GCM, is also proposed to implement various learning companions to satisfy the needs of various students.

1. Domain competency. The domain competency of a learning companion can be an expert, average student, or novice. The system must simulate how the learning companion solves problems. A learning companion, which is not an expert, may make mistakes or give erroneous recommendations. The following question arises: Can one implement the different domain competency levels of learning companions by setting and adjusting the attribute values of these learning companions? In GCM, simulating a learning companion's domain competency is the basis to simulate other characteristics of the learning companion.
2. Learning competency. A learning companion can learn when competing against and collaborating with the user or another learning companion. Having learned something, the learning companion should perform better or make fewer errors the next time on the same learning task. The system must determine "whether the learning companion will learn" and "when and how the learning companion learns."
3. Behavior of a peer tutor. When the user requests an opinion from the learning companion, the peer tutor can refuse such requests or provide various opinions. In addition, the peer tutor can provide opinions as an adaptive tutor, nonadaptive tutor, or evaluator. Therefore, the following question arises: How to simulate different behaviors of a learning companion in order to produce different peer tutors?
4. Behavior of a peer tutee. As a peer tutee, the learning companion solves problems, requests recommendations, and responds to the peer tutor's recommendations. In addition to solving a problem, the system must simulate "when the learning companion asks for recommendations" and "how the learning companion considers and responds to the peer tutor's different recommendations'."

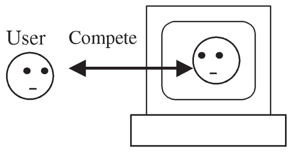


Fig. 1. Model one: Competitor without a collaborator.

The rest of this paper is organized as follows: Section 2 describes DwestAgent, an LCS implemented using the GCM approach. DwestAgent and its three learning models are used as illustrative examples to demonstrate the feasibility of GCM and to discuss the implementation requirements and related issues to simulate learning companions. Next, Sections 3, 4, 5, 6, and 7 present GCM's implementation scheme on steps and architecture, learning companions' domain competency, learning ability, behavior of a peer tutor, and behavior of a peer tutee, respectively. Section 8 describes a preliminary evaluation of the system. Conclusions are finally made in Section 9.

2 DwestAgent and Three Learning Models

DwestAgent, an LCS, is implemented using the GCM approach and in Java and Lisp. DwestAgent is used as an example not only to demonstrate the feasibility of GCM, but also to discuss the implementation requirements and related issues to simulate learning companions. DwestAgent provides three learning models: The user is a competitor without a collaborator, peer tutee, or peer tutor, respectively. In Distributed West, the user keys in a natural language to communicate with a human collaborator. In DwestAgent, the user communicates with the learning companion by using buttons or menus. The user also uses a calculator-like interface to compose the expression. These buttons, menus, and calculator-like interface provide the user with scaffolding tools to reduce the learning task complexity [10]. Scaffolding conventionally refers to the support that a teacher provides in helping a student implement a task [17]. While employed in a computer-based learning environment, scaffolding also refers to the tools that a computer offers. These scaffolding tools also reduce the complexity of simulating learning companions.

2.1 Model One: Competitor without Collaborator

In Model one, the user competes with a learning companion and with no collaborator (Fig. 1). This model resembles Model three of Distributed West. The difference is that the competitor in DwestAgent is a learning companion, not a human.

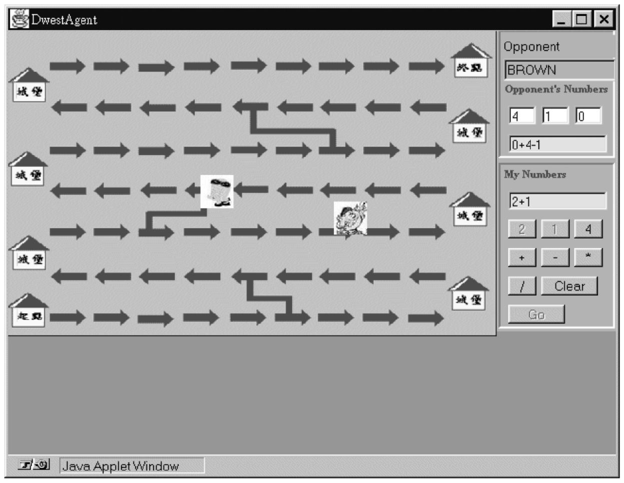


Fig. 2. Interface of model one.

The system randomly generates three numbers between zero and nine. These three numbers are displayed in three buttons. The usable operators are "+," "-," "×," and "/." The user pushes buttons representing numbers and operators to compose an expression (Fig. 2). The expression must include the three numbers exactly once. When the user pushes the button of a number, the number is added to the expression and the button is disabled to prevent the user from using the number again. The user can use an operator repeatedly, i.e., "1+2+3" is allowed. When the user completes an expression, the "Go" button is enabled. The user can then push the "Go" button and the outcome of the expression determines the user's move. The user and the competitor move in turns. The player reaching the final destination first wins the game. The user can see the numbers his/her opponent obtained and watch how his/her opponent compose an expression from those numbers. The "Clear" button is provided for the user to recompose an expression when the user has to.

The game has three special moves.

1. Shortcut. The board contains three shortcuts. A player moving onto a shortcut moves forward along the shortcut. The first shortcut is a jump from position six to position 15. The second one is a jump from 23 to 36. The third one is from 47 to 55.
2. Town. A player moving into a town jumps to the next town (10 more steps) as a bonus.
3. Bump. If a player moves to an opponent's exact location, the opponent is moved backward 10 steps.

2.2 Model Two: Peer Tutee

In Model two, the user collaborates with a learning companion to compete against another learning companion. The user functions as a peer tutee to control the buttons in order to compose an expression, while the learning companion collaborator functions as a peer tutor to provide recommendations (Fig. 3). In this model, the peer tutor, who is a learning companion, resembles a computer teacher when the peer tutor's competency is an expert. This model resembles Model two of Distributed West, except that the competitor is a learning companion instead of a human.

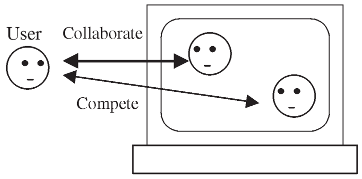


Fig. 3. Model two: peer tutee.

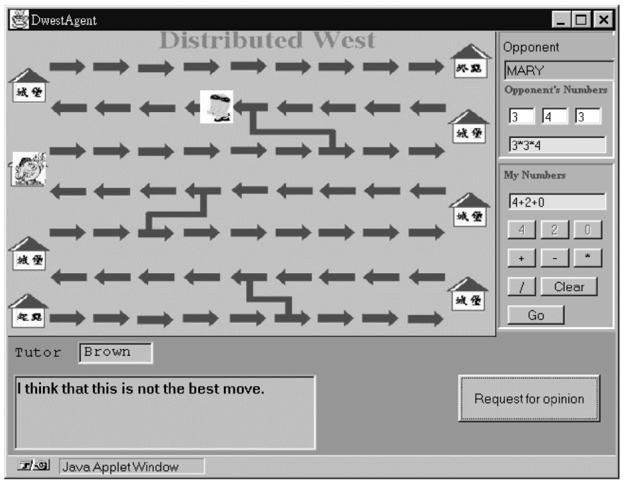
When the peer tutor's competency is not an expert, the peer tutor may give erroneous recommendations. Under this circumstance, the peer tutor resembles a troublemaker whose role is to interfere with the user deliberately as a strategy of "learning by disturbing" [12]. However, the troublemaker aims to make an error according to the user's confidence, while the peer tutor in DwestAgent acts based only on its domain competency and characteristics.

The interface of Model two resembles Model one (Fig. 4). The user pushes buttons to compose an expression. However, before acting, the user must request the peer tutor's recommendations at least once. This limitation allows the peer tutor to offer recommendations. Otherwise, the user can complete the game without requesting the peer tutor. The user can modify the expression based on the peer tutor's opinion, ask again, or act while neglecting the peer tutor's clue.

2.3 Model Three: Peer Tutor

In model three, the user collaborates with a learning companion to compete against another learning companion. The user functions as a peer tutor to guide the learning companion. Meanwhile, the learning companion functions as a peer tutee, which operates three numbers to compose an expression under supervision of the user (Fig. 5). In this model, the user learns by teaching the learning companion. This model resembles the "learning by teaching" approach investigated by other researchers whose model has no competitor [13], [14], [15].

The user observes the numbers and expressions, which the peer tutee makes. When the peer tutee requests a recommendation, the user chooses a recommendation from a menu and sends it to the peer tutee (Fig. 6). The user can



I think that this is not the best move.

Fig. 4. Interface of model two.

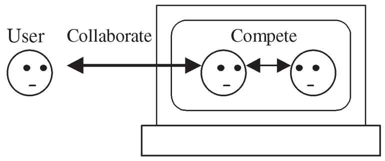


Fig. 5. Model three: Peer tutor.

respond only passively to a clue request of the peer tutee and cannot actively interrupt the peer tutee to offer recommendations. If the peer tutee asks again, the user chooses another recommendation.

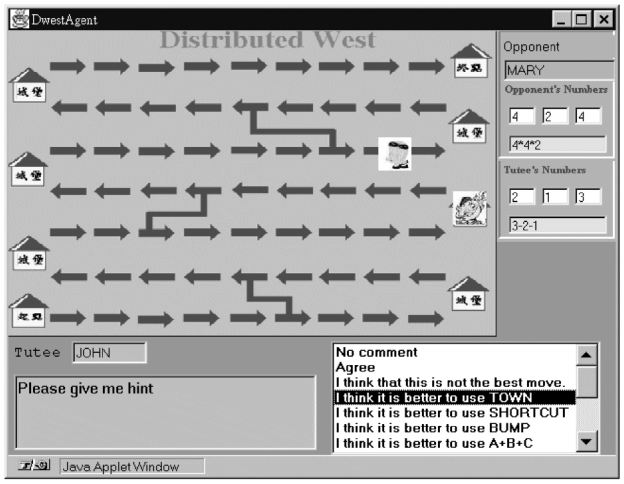
Opinion items on the menu are

1. No opinion or clue, e.g., “No comment.”
2. A positive comment, e.g., “Agree.”
3. A negative comment, e.g., “I think that this is not the best move.”
4. A clue with respect to an operator pattern, e.g., “I think it is better to use A+B*C.”
5. A clue with respect to a special move, e.g., “I think it is better to use Town.”
6. A direct recommendation with respect to a specific move or expression, e.g., “I think it is better to use 1+2×3.”

2.4 Variations of the Three Learning Models

DwestAgent provides three learning companions, i.e., John, Brown, and Mary, for the user to choose. They represent an expert, an average student, and a novice, respectively. Each learning companion can function as a competitor, peer tutor, or peer tutee. The user can select learning companions and their roles and then choose a learning model from the three give learning models.

The variations in learning companions’ roles and domain competency produce many variations of the three learning models. Table 1 classifies the DwestAgent’s learning companions according to their roles and domain competencies, and lists their similar agents in other systems. When functioning as a competitor, the learning companion is similar to learning companions of Contest-kids [18] and Distributed West. When functioning as a peer tutor and its proficiency is an expert, the learning companion is similar to the tutor of typical ITSs, strong companions of EduAgents [9], as well as the learning companion of RTS [10] and PALs [11]. When functioning as a peer tutor and its competency is not an expert, the learning companion is similar to weak companions of EduAgents, and the learning companion of Integration-kid [7], Troublemaker [12], and LuCy [8]. When functioning as a peer tutee and its proficiency is an expert, the learning companion is similar to a demo system. When functioning as a peer tutee and its proficiency is not an expert, the learning companion is similar to the agent of DENISE [14], STEPS [15], RTS, LECOBA [16], and PALs. In the following sections, DwestAgent is used as an example to present GCM.



Please give me hint

No comment
Agree
I think that this is not the best move.
I think it is better to use TOWN
I think it is better to use SHORTCUT
I think it is better to use BUMP
I think it is better to use A+B+C

Fig. 6. Interface of model three.

TABLE 1
Classification of Learning Companions of DwestAgent

Role \ Proficiency	Expert	Non-expert (Average student or Novice)
Competitor	Contest-kids	Contest-kids Distributed West
Peer tutor	ITSs RTS EduAgents PALs	Integration-kid Troublemaker LuCy EduAgents
Peer tutee	Demo system	STEPS DENISE RTS LECOBA PALs

3 Steps and Architecture

GCM uses six steps to simulate the characteristics of a learning companion. Steps one to three resemble each other with respect to constructing a typical ITS. Therefore, most techniques used in ITSs can be applied in GCM to implement LCSs.

1. Collect the required characteristic data of a set of students in their learning process. The data include problem solving states, paths, responses, and interactions;
2. Select an appropriate data representation to represent the user, i.e., a particular student;
3. Simulate an expert, including problem solving and interaction behavior;
4. Generalize the expert to simulate learning companions with different levels of domain competency and roles;
5. Initialize attribute values (parameters for simulation) of each learning companion to determine its characteristics and roles; and
6. Modify the attribute values of each learning companion to change itscharacteristics at an appropriate time.

GCM architecture consists of four main components: behavior module, domain module, user model, and learning companion pattern [19] (Fig. 7). The user model stores the user’s status observed by a system. The learning companion

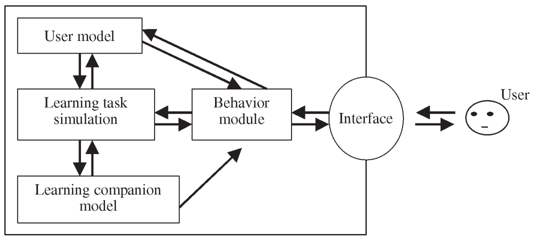


Fig. 7. GCM architecture.

pattern stores the learning companion's characteristic data. The domain module is responsible for simulating the domain competency of a learning companion. In addition, the behavior module is responsible for simulating the behavior of a learning companion. In DwestAgent, the domain module and the behavior module include sets of heuristic rules by taking the data of a learning companion pattern as parameters. Varying the data can produce various learning companions for different learning models. Setting multiple learning companion patterns can simulate multiple learning companions in an LCS.

4 Implementing the Domain Competency of Learning Companions

Simulating the domain competency of learning companions forms the basis of behavior simulation. The domain competency includes what a learning companion knows, the proficiency of a learning companion in domain related skills, and how a learning companion solves problems. The first two competencies can be established and stored in the learning companion pattern. GCM adopts the method of General Problem Solver (GPS) [20] and modifies it to satisfy the end of simulating problem solving competency of learning companions. Extensively applied to simulate human's problem solving behavior, GPS captures problem-solving behavior in a data structure such as problem states and operators. Moreover, GPS attempts to transform, with the appropriate operator(s), the current problem state into the goal state (Fig. 8). Many possible operators are available for a state to reach its neighboring states. The general heuristic selects the operator that reduces the difference between the current state and the goal state. GPS repeats the process until it reaches the goal state.

GPS has three procedures:

1. identify all neighboring states, which can be reached from the current state through the operators;

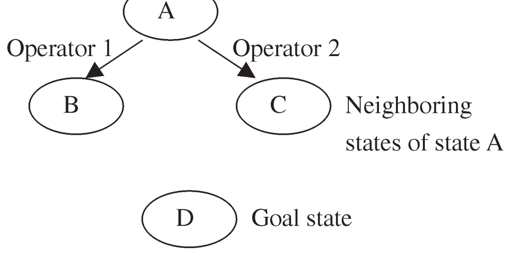


Fig. 8. Schematic diagram of General Problem Solver.

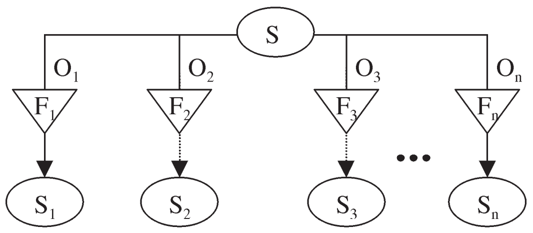


Fig. 9. Addition of a filter to neglect some states.

2. evaluate all these neighboring states; and
3. select the operator and transform the current state into the state with optimal evaluated value.

To simulate problem-solving processes of various learning companions, GCM makes two modifications on these three procedures. The modifications are based on two theories: overlay model and bug model, respectively. The theories are extensively used in user modeling. With different applications, the theories are applied herein to simulate learning companions. The underlying premise of the overlay model is that the knowledge or problem solving of a user is treated as a subset of that of an expert [21]. The theory is applied herein to simulate the problem solving of a learning companion by selecting a subset of expert problem solving. The bug model is a fixed collection of bugs and misconceptions to facilitate the detection of the status of users in ITSs [22]. This theory is applied herein to simulate the errors of the learning companion. If not proficient in some knowledge, the learning companion may fail when applying the knowledge.

4.1 Modification One: Applying the Overlay Model

The possible problem solving paths of a learning companion are a subset of those of an expert. A learning companion may neglect some possible operators of the current state owing to the learning companion's lack of proficiency with respect to the operators. If the learning companion neglects some operators, the first procedure does not consider some neighboring states. One or more proficiencies may be available for mastering an operator. Whether the learning companion neglects an operator depends on levels of corresponding proficiencies of the companion.

To implement modification one, a filter is added to each operator to determine whether a learning companion considers the operator. The levels of related proficiencies of a learning companion determine whether the operator will pass the filter. The inability of the operator to pass the filter implies that the learning companion neglects the operator. The dashed line arrow in Fig. 9 indicates that the learning companion neglects problem states S_2 and S_3.

An example of neglecting some states is given as follows: In a West game, the position of a learning companion is at five and its opponent is at 15 (Fig. 10). This situation is denoted as (5,15). The learning companion obtains three numbers, i.e., one, two, and three, from the computer. Assume that the level of the learning companion's proficiency with respect to operator pattern "1 + 2 + 3" is 0.8. The possibility of passing the filter is 80 percent, thus allowing the system to generate a random number to

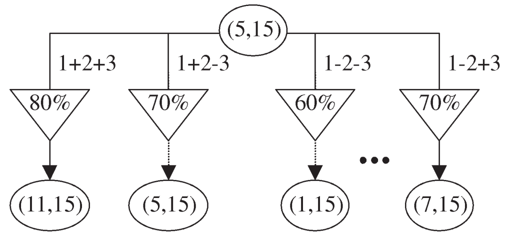


Fig. 10. An example of neglecting some states.

determine whether the learning companion considers the operator. In this case, the learning companion considers the operator "1 + 2 + 3."

4.2 Modification Two: Applying the Bug Model

A learning companion may incorrectly evaluate problem states in the second procedure and, therefore, choose either a suboptimal or wrong state. Whether a state is inaccurately evaluated depends on the related proficiencies of the learning companion.

To implement modification two, each neighboring state is given an evaluated value with respect to the difference between the state and goal state. Each state may have different possible evaluated values: the value evaluated by an expert and some other inaccurately evaluated values. These inaccurately evaluated values are generated because some related proficiencies of the learning companion are inadequate. Which evaluated value is selected for a state is calculated by the learning companion's related proficiencies with respect to evaluating the state. The problem state S_1 in Fig. 11 has two possible evaluated values: EV_1 and EV_2. The learning companion selects EV_1 as the evaluated value of problem state S_1.

An example of evaluating states is given as follows: In a West game, a learning companion's position is at five, and its opponent is at 15 (Fig. 12). The related proficiency with respect to correctly evaluating state (20,15) is "town." Assume that the level of the learning companion's proficiency with respect to "town" is 0.7. A 70 percent likelihood arises that the learning companion evaluates the state the same as an expert and a 30 percent likelihood to evaluate the state without considering "town." Based on this possibility, the system generates a random number to determine which evaluated value the learning companion takes. Another example is given as follows: The related proficiencies with respect to correctly evaluating state (6,15) are "shortcut" and "bump." In this case, the learning companion must consider "shortcut" first. The condition of

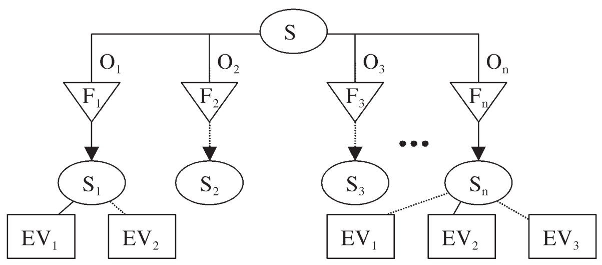


Fig. 11. Different evaulated values in a state.

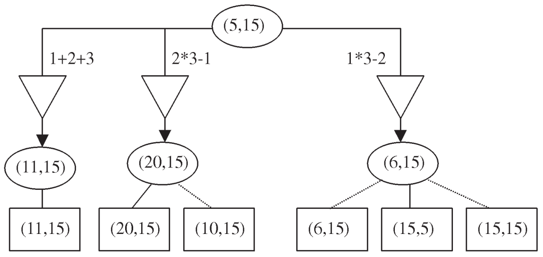


Fig. 12. An example of different evaluated values in a state.

"bump" can then be satisfied and may be considered. Assume that the levels of the learning companion's proficiencies with respect to "shortcut" and "bump" are 0.6 and 0.8, respectively. A 48 percent likelihood arises that the learning companion evaluates the state the same as an expert, 12 percent likelihood to evaluate the state considering "shortcut" and without considering a "bump," and 40 percent likelihood to evaluate the state without considering a "bump." Based on this possibility, the system generates a random number to determine which evaluated value the learning companion takes.

In the third procedure, the learning companion selects the neighboring state with the best-evaluated value to move. The strategy is to select a move, thus minimizing the distance from the goal and maximizing the distance the individual is ahead of an opponent. If several neighboring states have the same evaluated value, the learning companion must choose one to move. Two heuristics are proposed as follows:

1. Random. Randomly select one state from these neighboring states.
2. Favor. The learning companion may favor one of these states. Compare the likelihood of passing the operators, which reaches these neighboring states. Select the neighboring state with the highest possibility of passing the operator as its favorite state.

This novel approach to simulate the learning task allows one to take the learning companion pattern as parameters to simulate problem solving. Adjusting the data of the learning companion pattern varies the learning companion's problem solving competency and behavior. Additionally, multiple learning companion patterns can be established to yield several learning companions in a system.

5 Implementing Learning Competency of Learning Companions

A learning companion may learn from its competitor, peer tutor, and peer tutee. If learning something, the learning companion should improve next time on the same learning task. To implement the learning competency of a learning companion, the system must have mechanisms to determine when and how the learning companion learns. Two approaches are available to implement the learning competency of learning companions [7]: 1) applying machine learning methods to construct new knowledge or modify existing knowledge and 2) pretending or simulating to learn, i.e., the increase of the learning companion's knowledge or proficiencies is directly coded as part of the learning companion. DwestAgent adopts the simulation approach. The learning companion pattern stores the learning companion's proficiency levels in all special moves and operator patterns. Increasing the related proficiencies of a learning companion simulates the learning outcome of the learning companion. Each learning companion has three numbers to represent the learning competencies from competitor, peer tutee, and peer tutor, respectively. The learning competency is the increased amount of the related proficiencies when the learning companion learns. Setting a learning companion's learning competencies as zero implies that the learning companion does not learn anything.

When a competitor solves a problem, the learning companion also considers how to solve the problem if it is in the position of the competitor. Next, the learning companion compares its solution with the user's solution. If the user's solution is better, the system increases the learning companion's related proficiencies to simulate its learning. Related proficiencies include the operator pattern and tactics used in the better solution. Learning from a peer tutee resembles learning from a competitor.

When a peer tutor makes recommendations to a learning companion, the learning companion reconsiders its solution based on the tutor's recommendations. The learning companion learns whether if it discovers previous mistakes or confers with the tutor's recommendations. Section 7 proposes heuristics rules in terms of how a learning companion reconsiders, responds, and learns based on the tutor's various opinions.

6 Implementation of Peer Tutor

When functioning as a peer tutor, a learning companion must provide recommendations to the user. The learning companion can act as an evaluator, nonadaptive tutor, or adaptive tutor [6]. An evaluator simply evaluates the user's move. A nonadaptive tutor offers recommendations or strategies according to the user's current state and moves, whereas an adaptive tutor is also sensitive to the user's history. The following heuristic rules adhere to some of the principles, which are proposed by Burton and Brown to design a computer tutor [1].

6.1 Evaluator

An evaluator provides the tutee with either a positive or negative comment. In Heuristic Rule 1, the learning companion functions as an evaluator to effectively respond to the tutee's request for a recommendation.

Heuristic Rule 1: An evaluator's strategy.

1. Provide "no comment" when the user requests a recommendation without attempting any solution.
2. Provide a positive comment when the learning companion regards that the move of the user is the best.
3. Provide a negative comment when the learning companion regards that the move of the user is not the best.
4. Do not provide other recommendations even if the user requests again.

6.2 Nonadaptive Tutor

A nonadaptive tutor provides a tutee with recommendations based on the tutee's current status. In Heuristic Rule 2, the learning companion functions as a nonadaptive tutor, which provides opinions based on four successive levels of recommendations from general to specific. The learning companion provides a recommendation of level one for the first request. If the user asks again, the learning companion provides recommendations of level two or level three. Finally, the learning companion provides recommendations at level four.

Heuristic Rule 2: A nonadaptive tutor's strategy.

1. Level one. Provide a positive comment when the learning companion regards the move of the user is the best. Otherwise, provide a negative comment.
2. Level two. Provide a clue with respect to a special move if the best move uses a special move. Otherwise, provide clues at level three.
3. Level three. Provide a clue with respect to the operator pattern of the best move.
4. Level four. Provide a direct recommendation with respect to a specific expression of the best move.

6.3 Adaptive Tutor with a Domain-Based User Model

An adaptive tutor provides the tutee with recommendations, based not only on the tutee's current state, but also on the tutee's learning history. Each learning companion can construct its own user model or share the same user model [23]. The learning companion provides recommendations based on its user model with respect to the user. The user's domain weaknesses and independence status are modeled by the user model. The system then records the user's patterns of moves and used special moves. Next, the system compares these moves to the domain expert's move to indicate how many times the user used and missed some moves and special moves [1]. For example, the user uses operator pattern "A + B + C" and special move "shortcut," the system increases both the user's used times of pattern "A + B + C" and special move "shortcut." But, the expert uses pattern "A + B $\times$ C" and uses special move "town," then the system increases both the user's missed times of pattern "A + B $\times$ C" and special move "town." These used and missed times of patterns and special moves indicate the user's favor and weakness. In Heuristic Rule 3, the learning companion provides recommendations based on the user's domain weaknesses. The Heuristic Rule 3 is modified from Rule 2. However, when the best move uses a tactic, the learning companion takes the user's weaknesses into account.

Heuristic Rule 3: An adaptive tutor's strategy (with a domain-based user model):

1. Level one. Provide a positive comment when the learning companion regards that the move of the user is the best. Otherwise, provide a negative comment.
2. Level two. Provide a clue with respect to an operator pattern or a tactic according to the current state and the user's weakness.

3. Level three. Provide a direct suggestion with respect to a specific expression of the best move.

6.4 Adaptive Tutor with an Independence Model

The learning companion can also offer a recommendation based on the user's independence status. The independence status relates to the perceived feeling of needing, or not needing, as well as the tutor's assistance to accomplish the learning task [24]. The pedagogical strategy aims to increase the user's independence. Moreover, the system constructs an independence model of the user according to the interaction between the learning companions and the user.

Heuristic Rule 4: An adaptive tutor's strategy (with an independence model).

1. Choose "no comment" if the learning companion regards the user's independence as low.
2. Provide a recommendation based on Heuristic Rules 1, 2, or 3, otherwise.

According to Heuristic Rule 4, controlling the independence threshold of a learning companion can manipulate the recommendation-offering behavior of a learning companion. Each learning companion has its own independence threshold to determine whether the independence status of the user is high or low. A learning companion with a low independence threshold regards all users' independence as high, thus seldom or never rejecting request and always providing recommendations. A learning companion with a high independence threshold regards all users' independence as low and, therefore, does not provide recommendations.

In addition, the independence threshold can be accompanied by the independence modelling method to control the recommendation-offering behavior of a learning companion. For example, West adheres to the following principle: "*Do not tutor on two consecutive moves, no matter what*" [1]. A learning companion can adhere to this principle by setting the independence threshold of the learning companion to 0.5, adopting Heuristic Rule 4 as a recommendation-offering strategy, and adopting Heuristic Rule 5 as an effective means of modelling the user's independence. When the learning companion provides the user with recommendations, the system models the user's independence status as zero. The learning companion then regards the user's independence as low. Therefore, the learning companion rejects the user's request on the next move and does not tutor on two consecutive moves. When the learning companion rejects the user's request, the system models the user's independence status as one. The learning companion then regards the user's independence as high. Therefore, the learning companion tutors the user on the next move.

Heuristic Rule 5: Modeling the user's independence.

1. When the learning companion rejects the user's recommendation request, i.e., the learning companion gives "no comment," the system models the user's independence status as one.
2. When the user asks for a recommendation and the learning companion gives any comment or recommendation, the system models the user's independence status as zero.

The learning companion can model the user's independence by not only whether to offer recommendations or not, but also what to offer. The greater the detail that the learning companion provides implies a lower independence status of the user. For example, the system assigns a number from zero to one to model and represent the independence status of a user. Therefore, the system adopts Heuristic Rule 6 to model the user's independence.

Heuristic Rules 6: Modeling the user's independence.

1. When the learning companion rejects the user's recommendation request, the system increases the user's independence status by 0.5.
2. When the learning companion gives a positive comment, the system decreases the user's independence status by 0.1.
3. When the learning companion gives a negative comment, the system decreases the user's independence status by 0.2.
4. When the learning companion offers a recommendation with respect to an operator pattern or a special move, the system decreases the user's independence status by 0.4.
5. When the learning companion offers a recommendation with respect to a specific move or expression, the system decreases the user's independence status by 0.6.

The learning companion can adopt the "learning by disturbing" strategy to function as a troublemaker [12]. The learning companion intentionally gives an erroneous recommendation. Aimeur et al. [12] suggested the following: "*This strategy is useful only for learners who have already acquired a minimum amount of knowledge and should not be used on novices since it could discourage them.*" In addition, adopting the strategy may confuse the novices. The system should consider the user's confidence to determine when to interfere with the user.

7 Implementation of Peer Tutee

When a learning companion functions as a peer tutee, the system must simulate how it solves problems, when it requests a recommendation, and how it responds to the various hints from the peer tutor. In Section 4, we propose how to implement learning companion's problem solving capability. The learning companion asks the peer tutor for recommendations in the following situations:

1. The learning companion does not know what to do. The only option available to the learning companion is to ask the peer tutor for assistance.
2. The learning companion is required to ask the peer tutor for an recommendation at least once after completing a expression.
3. The learning companion who does not understand the peer tutor's recommendation asks the peer tutor for clarification.

The peer tutor may provide various recommendations, such as "no comment," a positive comment, a negative comment, a clue with respect to an operator pattern, a clue with respect to a special move, or a direct recommendation with respect to a specific expression or move. The learning companion reconsiders and learns based on the tutor's various recommendations. The learning companion can then change its process, request again, or continue its previous process. The learning companion learns something if it finds that it has made a mistake or previously neglected something. The following is a set of heuristic rules for different recommendations.

Heuristic Rule 7: How a learning companion responds to various recommendations.

1. If the tutor's recommendation is a positive comment or no comment, the learning companion continues with its process.
2. If the tutor's recommendation is a negative comment, the system runs the learning task simulation again, i.e., the learning companion reconsiders.
3. If the tutor's recommendation is one with respect to an operator pattern, the system temporarily increases the level of proficiency with respect to the pattern and runs the learning task simulation again. Therefore, the possibility of passing those operators with this proficiency increases. Restated, the learning companion reconsiders and pays closer attention to those operators with the proficiency that the tutor recommends at this time.
4. If the tutor's recommendation is one with respect to a special move (A special move affects evaluated value of a state), the system temporarily increases the level of proficiency with respect to evaluating the state and runs the learning task simulation again. Therefore, the possibility of correctly evaluating those states with this proficiency increases. Restated, at this time, the learning companion reconsiders and more closely evaluates those states with this proficiency.
5. If the tutor's recommendation is a direct one with respect to a specific expression or move, the system eliminates the filter on this operator. Therefore, the neighboring state that the operator reaches is evaluated. Restated, the learning companion reconsiders and does not neglect this operator, and its reaching neighboring state is considered and evaluated.

When a learning companion reconsiders its move, the learning companion compares the outcome with the previous one to determine whether to change its process or continue its previous process. If the second outcome is better than the first one, the learning companion changes its process and the system increases the levels of the learning companion's proficiencies relating to the error during first attempt to solve the problem. That is, the learning companion detects something erroneous or neglected and pays closer attention to similar cases in the following process. Otherwise, the learning companion continues with its previous process.

The learning companion has two options when not comprehending why the tutor makes such a recommendation: It asks again or neglects the tutor's opinion. In Heuristic Rule 7, the learning companion asks again. Some other possible heuristics are shown as follows:

TABLE 2
Students Preferred Learning Companions in Three Learning Models

	Expert	Mediocrity	Novice
Model one (Competitor)	23	11	6
Model two (Tutor)	27	6	5
Model three (Tutee)	16	10	9

1. Random. The learning companion randomly decides to ask again or neglect the tutor's opinion. Varying the ratio of possibility can alter the learning companion's behavior.
2. Independence. The learning companion asks again when its independence is low. Otherwise, it neglects the tutor's opinion. Each learning companion has its own independence status. Controlling the learning companion's independence status can manipulate its behavior.

8 Preliminary Evaluation

A preliminary evaluation was conducted to investigate which learning model the students preferred, which competence level of a learning companion the students preferred, and whether learning companions are humanlike or not. The subjects of this evaluation were 42 students of the computer science department at Jin-Wen Institute of Technology. The students engaged in the three learning models of DwestAgent with three different learning companions and then completed a questionnaire. Table 2 shows students' preferred learning companions in three learning models.

The evaluation of DwestAgent was conducted in three periods. In the first period of evaluation, the students played Model one of DwestAgent three times to compete against learning companion John (expert), Brown (mediocrity), and Mary (novice), respectively. Twenty-three students preferred the expert as their competitor. Most of them stated that it was challenging to compete against an expert. Eleven students preferred the mediocre competitor. They stated that the competence level of the competitor is about the same with them. Six students preferred the novice competitor because they could defeat the competitor to attain a sense of achievement.

Investigation into the relationship between game results and students' preferred opponents shows that, 13 students, who were defeated by the expert but defeated other learning companions, preferred the expert competitor. On the contrary, another 13 students, while attaining the same game result, preferred other learning companions. We attribute the investigation result to personalities of students. Some students prefer more challenging, while some prefers less challenging.

In the second period, the students played Model two of DwestAgent three times under the advice of John, Brown, and Mary, respectively. Twenty-seven students preferred the expert tutor because the opinions of this tutor were perfect and insightful. The students who preferred mediocre or novice tutors stated that their preferred tutor did not always criticize their solutions so that they could maintain a better sense of achievement.

TABLE 3
Student's Preferred and Most Benefited Learning Models

	Model one	Model two	Model three
Preferred	30	10	2
Benefited	21	9	10

In the third period, the students played Model three of DwestAgent three times to tutor John, Brown, and Mary, respectively. Sixteen students preferred the expert tutee because the tutee's moves were perfect and they could learn from it. Ten students preferred the mediocre tutee because the domain competence of this learning companion was about the same as theirs. Nine students, who preferred the novice tutee, felt superior because their moves were better than the novice tutee.

Among the three learning models, most students (see Table 3) preferred model one, or competing against learning companions. The reason is that they have total control on their moves. Ten students preferred having a tutor to provide opinions. While answering in which learning model the students could learn more, 21 students chose model one because they must think by themselves and cannot depend on any learning companion. Nine students chose model two because the tutor could suggest alternative solutions, thus provoking them to think more. Ten students chose model three because observing how the tutee made moves lended them opportunities to learn different solutions.

Two-thirds of the students regarded the learning companions not humanlike (Table 4). The reasons stated by these students can be classified as following: They are too perfect, too stupid, working in a fixed mode, lacking representations of human beings, and others. The shortcomings of being too perfect or too stupid can be improved by adjusting learning companion patterns. Employing more heuristic rules or randomness can introduce more working modes of the learning companions. Adding voice and animation to the learning companions can enrich representations of human beings. However, one third of students regarded the learning companion humanlike. In addition, one of the purposes of the learning companions is to provide students with a social learning environment. Whether these learning companions are humanlike or not is just a factor of the students' motivation.

9 Conclusions

This study proposes an approach, named GCM, to implement learning companions with different domain competency and different roles. DwestAgent, an LCS implemented in GCM, provides an example to demonstrate the feasibility of GCM and to discuss the implementation requirements and related issues in simulating the domain competency, learning competency, and behavior as a peer tutor, and behavior as a peer tutee of a learning companion. GCM takes data of learning companion patterns as parameters to simulate the learning companions. With different values of these parameters, learning companions possess different competency and behaviors. Varying the data of learning companion patterns can alter its competency and behavior. Moreover, a learning companion system simultaneously simulates several learning companions by establishing several learning companion patterns.

TABLE 4
Whether Learning Companions are Humanlike

	Humanlike	Not humanlike
Model one (Competitor)	12	29
Model two (Tutor)	13	26
Model three (Tutee)	14	26

The paper also shows the variations of three learning models in DwestAgent. These variations are produced by employing learning companions of different competency levels. These variations and their similar learning companion systems, which are designed in different theories by other researchers, are listed. Such variation shows the strengths and goals of GCM: Varying the role of a learning companion and adjusting the data in the learning companion pattern produces various learning companions for different learning models and different theories.

In addition, a preliminary evaluation was conducted to investigate which learning model the students preferred, which competency level of a learning companion the students preferred, and whether the learning companions were humanlike or not. The evaluation results show that the students' preferred learning models and learning companions vary. Thus, a flexible and adaptive approach like GCM to implement learning companions with different roles and domain competencies is essential to satisfy students' needs.

Applications implemented in GCM are available to students to choose their preferred learning companions and learning models. Another application of GCM is to design a series of games with different learning companions and different game maps. The series should include practicing arithmetic skills step by step. Additionally, scaffold-fading strategies should also be applied to design learning companions. That is, the user should compete against a "weak" learning companion and collaborates with a "strong" learning companion initially. Later, the competitor should become stronger and stronger, while the collaborator becomes weaker and weaker. This design allows the student to receive gradually decreased help from the learning companions and to face gradually increased challenges. In addition, students can defeat the "weak" learning companion to attain a sense of achievement and then take the challenge of next stage.

Acknowledgments

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC89-2520-S-008-011. The authors would also like to thank Yih-Ruey Juang and the students of the Jin-Wen Institute of Technology for their participation in the evaluation.

References

[1] R.R. Burton and J.S. Brown, "An Investigation of Computer Coaching for Informal Learning Activities," *Int'l J. Man-Machine Studies,* vol. 11, pp. 5-24, 1979.
[2] J. Self, "A Perspective on Intelligent Computer-Assisted Learning," *J. Computer Assisted Learning,* vol. 1, pp. 159-166, 1985.

[3] A. Gilmore and J. Self, "The Application of Machine Learning to Intelligent Tutoring Systems," *Artificial Intelligence and Human Learning, Intelligent Computer-Aided Instruction,* J. Self, ed., pp. 179-196, New York: Chapman and Hall, 1988.

[4] T.W. Chan and A.B. Baskin, "Studying with the Prince: The Computer as a Learning Companion," *Proc. Int'l Conf. Intelligent Tutoring Systems (ITS '88),* pp. 194-200, 1988.

[5] P. Dillenbourg and J. Self, "People Power: A Human-Computer Collaborative Learning System," *Proc. Second Int'l Conf. Intelligent Tutoring Systems,* C. Frasson, G. Gauthier, and G. McCalla, eds., pp. 651-660, 1992.

[6] T.W. Chan, Y.L. Chung, R.G. Ho, W.J. Hou, and G.L. Lin, "Distributed Learning Companion Systems—West Revisited," *Proc. Second Int'l Conf. Intelligent Tutoring Systems,* C. Frasson, G. Gauthier, and G. McCalla, eds., pp. 643-650, 1992.

[7] T.W. Chan and A.B. Baskin, "Learning Companion Systems," *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education,* chapter 1, C. Frasson and G. Gauthier, eds., pp. 6-33, New Jersey: Ablex Publishing Corp., 1990.

[8] B. Goodman, A. Soller, F. Linton, and R. Gaimari, "Encouraging Student Reflection and Articulation Using a Learning Companion," *Int'l J. Artificial Intelligence in Education,* vol. 9, pp. 237-255, 1998.

[9] P. Hietala and T. Niemirepo, "The Competency of Learning Companion Agents," *Int'l J. Artificial Intelligence in Education,* vol. 9, pp. 178-192, 1998.

[10] T.W. Chan and C.Y. Chou, "Exploring the Design of Computer Supports for Reciprocal Tutoring," *Int'l J. Artificial Intelligence in Education,* vol. 8, pp. 1-29, 1997.

[11] L.A. Scott and F. Reif, "Teaching Scientific Thinking Skills: Students and Computers Coaching Each Other," *Proc. Ninth Int'l Conf. Artificial Intelligence in Education (AI-ED 99),* pp. 285-293, 1999.

[12] E. Aimeur, H. Dufort, D. Leibu, and C. Frasson, "Some Justifications for the Learning by Disturbing Strategy," *Proc. World Conf. Artificial Intelligence in Education (AI-ED '97),* pp. 119-126, 1997.

[13] S. Palthepu, J. Greer, and G. McCalla, "Learning by Teaching," *Proc. Int'l Conf. Learning Sciences,* pp. 357-363, 1991.

[14] D. Nichols, "Issues in Designing Learning by Teaching Systems," AAI/AI-ED Technical Report No. 107, Computing Department, Lancaster Univ., Lancaster, United Kingdom, 1994.

[15] K. VanLehn, S. Ohlsson, and R. Nason, "Applications of Simulated Students: An Exploration," *J. Artificial Intelligence in Education,* vol. 5, no. 2, pp. 135-175, 1994.

[16] R. Uresti and J.A. Lecoba, "A Learning Companion System for Binary Boolean Algebra," *Proc. Workshop 1: Animated and Personified Pedagogical Agents, AI-ED'99 Conf.,* L. Johnson, ed., pp. 56-61, 1999. Also available in *Int'l J. Artificial Intelligence in Education,* vol. 10, pp. 1060-1069, 1999.

[17] A. Collins, J.S. Brown, and S.E. Newman, "Cognitive Apprenticeship: Teaching the Craft of Reading, Writing, and Instruction," *Essays in Honor of Robert Glaser,* Hillsdale, N.J.: Lawrence Erlabum Associates Publishers, 1989.

[18] T.W. Chan and J.A. Lai, "Contest-Kids: A Competitive Distributed Social Learning Environment," *Proc. World Conf. Computers in Education,* pp. 767-776, 1995.

[19] C.Y. Chou and T.W. Chan, "Redefining Learning Companions: Past, Present, and Future of Educational Agents," submitted for publication.

[20] A. Newell and H.A. Simon, "GPS, A Program That Simulates Human Thought," *Computers and Thought,* E.A. Feigenbaum and J. Feldman, eds., pp. 279-293, New York: McGraw-Hill, 1963.

[21] B. Carr and I.P. Goldstein, "Overlays. A Theory of Modeling for Computer-Aided Instruction," AI Lab Meno 406, MIT, Cambridge, Mass., 1977.

[22] J.S. Brown and R.R. Burton, "Diagnostic Models for Procedural Bugs in Basic Mathematical Skills," *Cognitive Science,* vol. 2, pp. 155-191, 1978.

[23] C.Y. Chou, C.J. Lin, and T.W. Chan, "User Modeling in Simulating Learning Companions," *Ninth Int'l Conf. Artificial Intelligence in Education (AI-ED '99),* pp. 277-284, 1999.

[24] A. del Soldato and B. du Boulay, "Implementation of Motivational Tactics in Tutoring Systems," *J. Artificial Intelligence in Education,* vol. 6, no. 4, pp. 337-378, 1995.

Chih-Yueh Chou received the PhD degree in computer science and information engineering from the National Central University, Taiwan, in 2000. He is a postdoctor in the Learning Technology Research Center of National Central University. His current research interests are intelligent distance learning, social learning theory and systems, and intelligent educational agents.

Tak-Wai Chan received the PhD degree in computer science from the University of Illinois at Urbana-Champaign, in 1989. He is a professor in the Department of Computer Science and Information Engineering at National Central University, Taiwan. His current research interests are intelligent distance learning, social learning theory and systems, and intelligent educational agents. He is an associate editor of the *International Journal of Educational Telecommunication*.

Chi-Jen Lin received the MS degree in computer science and information engineering from the National Central University, Taiwan in 1995. He is a PhD student in the Department of Computer Science and Information Engineering at National Central University. His current research interests include distant learning, artificial intelligence in education, and intelligent agents.