2012

Leveraging Help Requests In Pomdp Intelligent Tutors

Jeremiah Folsom-Kovarik

University of Central Florida

Part of the Computer Sciences Commons, and the Engineering Commons

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

https://stars.library.ucf.edu/etd/2262
LEVERAGING HELP REQUESTS
IN POMDP INTELLIGENT TUTORING SYSTEMS

by

JEREMIAH T. FOLSOM-KOVARIK
M.S. University of Central Florida, 2008
B.A. Drew University, 2000

A dissertation offered in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2012

Major Professors: Gita Sukthankar
Sae Schatz
© 2012 Jeremiah T. Folsom-Kovarik
ABSTRACT

Intelligent tutoring systems (ITSs) are computer programs that model individual learners and adapt instruction to help each learner differently. One way ITSs differ from human tutors is that few ITSs give learners a way to ask questions. When learners can ask for help, their questions have the potential to improve learning directly and also act as a new source of model data to help the ITS personalize instruction. Inquiry modeling gives ITSs the ability to answer learner questions and refine their learner models with an inexpensive new input channel.

In order to support inquiry modeling, an advanced planning formalism is applied to ITS learner modeling. Partially observable Markov decision processes (POMDPs) differ from more widely used ITS architectures because they can plan complex action sequences in uncertain situations with machine learning. Tractability issues have previously precluded POMDP use in ITS models. This dissertation introduces two improvements, priority queues and observation chains, to make POMDPs scale well and encompass the large problem sizes that real-world ITSs must confront.

A new ITS was created to support trainees practicing a military task in a virtual environment. The development of the Inquiry Modeling POMDP Adaptive Trainer (IMP) began with multiple formative studies on human and simulated learners that explored inquiry modeling and POMDPs in intelligent tutoring. The studies suggest the new POMDP representations will be effective in ITS domains having certain common characteristics.
Finally, a summative study evaluated IMP’s ability to train volunteers in specific practice scenarios. IMP users achieved post-training scores averaging up to 4.5 times higher than users who practiced without support and up to twice as high as trainees who used an ablated version of IMP with no inquiry modeling. IMP’s implementation and evaluation helped explore questions about how inquiry modeling and POMDP ITSs work, while empirically demonstrating their efficacy.
To my loving family
ACKNOWLEDGMENTS

This work is supported in part by the Office of Naval Research Grant N0001408C0186, the Next-generation Expeditionary Warfare Intelligent Training (NEW-IT) program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the ONR or the US Government. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

This dissertation owes much to the support and direction of my co-advisors, Dr. Sae Schatz and Dr. Gita Sukthankar. I thank them both for allowing me great latitude in exploring my personal interests and for suggesting boundaries and requirements that kept me moving forward. I learned a lot from both of you and I’m grateful that you could work together in advising me. I also thank my other dissertation committee members, Dr. Avelino Gonzalez and Dr. Randall Shumaker, for their valuable feedback, questions, and criticism that helped to hone and clarify my work. I thank Dr. Jennifer Vogel-Walcutt for her help with designing materials for the human-participant studies related to this dissertation. I thank my fellow student Mr. Robb Dunn for the many productive discussions over the time we have worked together. I additionally thank the UCF Department of Electrical Engineering and Computer Science and the UCF Institute for Simulation and Training, especially Dr. Denise Nicholson and Dr. Stephanie Lackey, for supporting me during my time as a UCF student and for furnishing a focused and well rounded education and background.
Finally, I thank my lovely wife Jun for inspiring me to complete this dissertation and teaching me that hard work and persistence will pay off. Thank you!
TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................................... xiv

LIST OF TABLES ............................................................................................................................... xix

LIST OF ABBREVIATIONS ............................................................................................................... xxiii

CHAPTER 1: INTRODUCTION .......................................................................................................... 1

1.1 Intelligent Tutoring Systems for Effective and Efficient Instruction ................................. 1

1.2 Inquiry Modeling ...................................................................................................................... 3

1.3 POMDPs .................................................................................................................................. 5

1.3.1 POMDP Definition ............................................................................................................... 5

1.3.2 POMDPs in ITSs .................................................................................................................. 10

1.4 Contributions and Research Hypotheses .............................................................................. 13

1.4.1 Contributions of this Dissertation ..................................................................................... 13

1.4.2 Inquiry Modeling Hypothesis ............................................................................................. 14

1.4.3 Scaling Hypothesis ............................................................................................................. 15

1.4.4 Planning Hypothesis .......................................................................................................... 15

1.4.5 Summative Hypothesis ...................................................................................................... 16

1.4.6 Published Works ................................................................................................................ 16

CHAPTER 2: SURVEY OF RELATED WORK ................................................................................. 19

2.1 Intelligent Tutoring System User Models .............................................................................. 19
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 POMDP Observation Compression</td>
<td>77</td>
</tr>
<tr>
<td>3.2.1 Example User Model Continuation</td>
<td>77</td>
</tr>
<tr>
<td>3.2.2 Problem Characteristics and the Observation Chain Representation</td>
<td>79</td>
</tr>
<tr>
<td>3.3 POMDP ITS Representation Summary</td>
<td>82</td>
</tr>
<tr>
<td>CHAPTER 4: THE CFF INSTRUCTIONAL DOMAIN</td>
<td>84</td>
</tr>
<tr>
<td>4.1 FOPCSIM</td>
<td>85</td>
</tr>
<tr>
<td>4.2 Use Case Tasks</td>
<td>88</td>
</tr>
<tr>
<td>4.3 Call for Fire Summary</td>
<td>89</td>
</tr>
<tr>
<td>CHAPTER 5: FORMATIVE STUDIES</td>
<td>91</td>
</tr>
<tr>
<td>5.1 Expanded and Updated ITS Development Cost Information</td>
<td>91</td>
</tr>
<tr>
<td>5.1.1 Method</td>
<td>92</td>
</tr>
<tr>
<td>5.1.2 Results</td>
<td>93</td>
</tr>
<tr>
<td>5.1.3 User Model Cost Discussion</td>
<td>96</td>
</tr>
<tr>
<td>5.2 POMDP ITS Simulation Study</td>
<td>102</td>
</tr>
<tr>
<td>5.2.1 Method</td>
<td>102</td>
</tr>
<tr>
<td>5.2.2 Experimental Setup</td>
<td>106</td>
</tr>
<tr>
<td>5.2.3 Results</td>
<td>108</td>
</tr>
<tr>
<td>5.2.4 Simulation Study Discussion and Summary</td>
<td>114</td>
</tr>
<tr>
<td>5.3 Inquiry Modeling Observational Study</td>
<td>117</td>
</tr>
</tbody>
</table>
6.5 Conclusion ........................................................................................................ 164

CHAPTER 7: SUMMATIVE EVALUATION ................................................................. 165

7.1 Method ............................................................................................................... 165

7.1.1 Materials and Procedures ............................................................................. 165

7.1.2 Treatments ...................................................................................................... 168

7.1.3 Instruments .................................................................................................... 168

7.1.4 Hypotheses ..................................................................................................... 171

7.2 Results ................................................................................................................ 173

7.2.1 Participants .................................................................................................... 173

7.2.2 Proficiency Tests ........................................................................................... 176

7.2.3 Written Knowledge Tests .............................................................................. 180

7.2.4 User Acceptance ........................................................................................... 181

7.2.5 Tutor Interaction Patterns .............................................................................. 186

7.2.6 Tutor Help Efficacy ...................................................................................... 191

7.2.7 Tutor Belief Accuracy ................................................................................... 193

7.2.8 Summary of Results ..................................................................................... 198

7.3 Discussion .......................................................................................................... 200

7.3.1 POMDPs ...................................................................................................... 201

7.3.2 POMDP Lessons Learned ............................................................................ 206
LIST OF FIGURES

Figure 1: Intelligent tutoring systems are a subset of computer learning environments that “close the loop,” internally modeling users to determine the best way to help them learn. 2

Figure 2: The POMDP representation aligns well with traditional ITS tutoring tasks and process flow (compare with Figure 1). After the ITS is constructed in advance (a), each interaction with a user updates the tutor’s user model (belief) and suggests the next action the system should take (b). 6

Figure 3: Freedom. Different question interfaces may have an impact on the information questions contain and on user experience, including cognitive load. 47

Figure 4: Model integration. Questions can update global and domain-specific learner model data. 51

Figure 5: Response roles. An ITS can take on one or more social roles when responding to questions. 54

Figure 7: A schematic representation of the FOPCSIM interface. A realistic image fills the screen, and overlaying it are icons to activate simulated tools. Activating the radio tool causes several text boxes and menus to overlay and obscure the screen. Trainees interact with the simulator using a keyboard and mouse (the arrow pointer at right). 86

Figure 8: Direct comparison to a lossless POMDP on problems with $|K| \leq 8$, $a = 0.5$, $d = 0.5$ shows the experimental representations did not substantively degrade tutoring performance. 109

Figure 9: When $|K| = 8$, $|A| = 8$, and $|\Omega| = 64$, as here, lossless POMDP performance (a) varies with action efficacy, not priority effect. Experimental POMDPs perform about the
same as lossless in many conditions (b, c, d), but state queues substantively degrade
performance at some points (circled), indicating they are unsuitable for priority effect
d < 0.25. .................................................................

Figure 10: Performance on large problems can be improved up to approximately double,
if every observation contains information about multiple features. Observation chaining
greatly increases the number of independent dimensions a single observation can
completely describe (the number above each column). ........................................ 114

Figure 11: Simply tutoring the most recently observed gap (a) causes poor performance
when gap priority is important. POMDP scores (see Figure 9a) are a fraction of the
comparison scores at all points (b). The POMDP performs substantively better at many
points (circled). ........................................................................................................ 116

Figure 12: Trainee performance improved on several measures during the human study.
....................................................................................................................................... 132

Figure 13: Affective states trainees experienced during simulations varied, suggesting
usefulness for a user model. ......................................................................................... 134

Figure 14: IMP’s trainee-facing screen layout consists of a question menu that is always
open at the screen bottom and question answers or IMP-initiated hints that appear in the
screen center and can be moved, resized, or closed. Behind the IMP windows is the full-
screen simulator window, with relative size and position indicated by a white rectangle.
....................................................................................................................................... 138

Figure 15: An example of an IMP-initiated intervention, displaying the elaborated version
of the RF hint. ............................................................................................................. 139
Figure 16: Using a menu interface for the QUI both lets trainees choose which questions they want to ask IMP and also implicitly suggests questions that trainees might not have considered otherwise................................................................. 141

Figure 17: IMP’s experimenter interface allows basic configuration changes........... 144

Figure 18: IMP component interactions. Rectangles represent components, with shaded rectangles indicating components that are specific to the CFF domain and white rectangles indicating generic, reusable components. Arrows indicate control flow during training. Double-headed arrows indicate calls to the APPL generic POMDP library.... 145

Figure 19: Policy search on the two IMP models progressed over the entire 48 hours allowed it. Both policies grew to about 2,000 alpha vectors, letting IMP make about 2,000 different decisions based on learner interactions. Policy search visited fewer points in belief space with the full IMP model because its greater complexity slowed the search. ....................................................................................................................... 162

Figure 20: During policy search, reward estimates improved quickly for a few seconds and slowly thereafter. However, the small improvements might not mean useful search ended, since policy complexity was still increasing (Figure 19). The time axis is on a log scale to emphasize the first seconds........................................................................................................ 163

Figure 21: For both simulator tests, participants in different conditions did not differ significantly in their speed of issuing CFFs or in their rate of hitting enemy targets or friend targets. Therefore any tutor interruptions during practice did not cause a significant impact on automaticity or proficiency in using the CFF simulator. Error bars indicate 95% confidence intervals................................................................. 177
Figure 22: Participants in the P and Q conditions demonstrated significantly better target selection than the control participants. In target engagement, all conditions were significantly different, with P better than the control and Q better than P. Error bars indicate 95% confidence intervals. ................................................................. 179

Figure 23: Participants in the Q condition significantly outscored participants in the P condition and control participants, who scored about the same. Although there was no difference between conditions on extension test items requiring application to new situations, the Q participants showed significantly improved recall of knowledge directly related to the practice. Error bars indicate 95% confidence intervals......................... 181

Figure 24: Nine user acceptance survey items that showed the most significant differences between conditions in the number of participants who agreed or strongly agreed. Items are ordered by their introduction in the text. For each item, the three horizontal bars correspond to the three experimental conditions, and are divided according to the distribution of participant responses. Colored bars highlight the percentage of participants who agreed and strongly agreed with each item (or disagreed and strongly disagreed with reversed items)................................................................. 186

Figure 25: Hints that IMP chose for more than one participant in the P condition. Oval leaves represent several diverging choices that were different for every participant (label indicates number of paths). By the fifth turn, IMP’s personalization was responsive enough to present different material to every participant except for two pairs of people. ......................................................................................................................... 189
Figure 26: Hints that IMP chose for more than one participant in the Q condition. Oval leaves represent several diverging choices that were different for every participant (label indicates number of paths). Tutor interactions diverged more quickly than in the P condition, suggesting learner questions offered more observations to help IMP personalize its interactions. However, one group of participants received the “Hurry” hint several times in a row. ................................................................. 190

Figure 27: In both experimental conditions, IMP presented most participants with unique interventions rather than interventions that it showed to any other participant. In the Q condition, IMP’s decision paths diverged more quickly than in the P condition. .......... 191

Figure 28: Performance during practice of the participants IMP estimated were done at the end of practice (top group), compared to those IMP estimated still had knowledge gaps at the end of practice (bottom group). IMP’s knowledge estimates correctly reflected significant differences between groups during practice. However, the model did not distinguish between learners in a finer gradation than “Done” or “Not Done.”.......... 195

Figure 29: Participant retrospection of affective states was compared to IMP estimates. In all cases where IMP estimates differed between groups, shown here, the direction of the difference was correct. Black markers indicate significant differences, while light grey markers are shown for completeness. Error bars indicate 95% confidence intervals, and have horizontal width proportional to the number of participants in each group.......... 198
LIST OF TABLES

Table 1: ITS costs and outcomes by learner model type based on published literature review ................................................................. 43

Table 2: Individual reports of macroadaptation models’ development cost in relation to ITS teaching time ................................................................. 95

Table 3: Individual reports of microadaptation models’ development cost in relation to ITS teaching time ................................................................. 95

Table 4: Individual reports of an entire ITS’s development cost in relation to its teaching time, showing models used ................................................................. 96

Table 5: Best-case scenario model costs, as determined by finding the lowest cost ratio reported for each model category ................................................................. 98

Table 6: Costs and effects on learning of different model types. Compare to Table 1 . 101

Table 7: Performance of compression schemes as compared to a lossless representation. Figure 9 visualizes this data ................................................................. 112

Table 8: Levels of cognitive load trainees might experience during training or performance ................................................................. 119

Table 9: Affective states trainees might experience during scenarios, with the definitions presented to study participants ................................................................. 120

Table 10: KSAs required for good performance in the simplified CFF domain ............ 121

Table 11: Help seeking event taxonomy ................................................................. 124

Table 12: Observational study agenda ................................................................. 127

Table 13: Help request feature observation frequency ................................................................. 130
Table 14: New knowledge gaps that participants demonstrated......................... 133

Table 15: The 18 knowledge gaps in IMP’s learner model are listed in state queue order.
.............................................................................................................................. 150

Table 16: Example text of hints that address each gap. IMP has at least a brief hint and an elaborated hint to address each modeled gap........................................................................... 152

Table 17: Observations IMP can use to infer a trainee’s current knowledge state.
Observation chains let IMP represent observations containing several dimensions of additional information or multiple values. Each observation gives evidence for or against the presence of the related knowledge gaps........................................................................... 154

Table 18: Observations IMP can use to infer a trainee’s current cognitive state. Plus (+) means a state is more likely and minus (−) means a state is less likely after the observation, while 0 indicates no change except for relative changes. ...................... 155

Table 19: Questions trainees can ask IMP. Menu refers to the display group tab where the question appears in the QUI, and during simulator usage training IMP only displays the Radio Menu and Training menu tabs. Topic refers to which set of knowledge gaps IMP makes inferences about when a trainee asks a question. ........................................... 156

Table 20: Summative study agenda.................................................................................. 167

Table 21: Experimental design for IMP’s evaluation....................................................... 168

Table 22: Results of testing whether IMP’s affective state estimates differed between groups of participants who actually felt the same state more or less frequently. Eight of sixteen differences were statistically significant. Post-hoc tests (third line in cells with
significant differences) show that where differences existed, they were in the correct direction. ................................................................................................................................. 196

Table 23: Overall, the study showed that IMP improved training. Inquiry modeling produced better results than an ablated IMP without inquiry modeling. The POMDP learner model reflected trainees’ knowledge and affective states................................. 200

Table 24: A transcript of a typical training session in the P condition demonstrates how the POMDP learner model lets IMP address likely knowledge gaps before it directly observes them, choose which gap to address when it observes several, ignore mistakes that may be slips, and generally intervene more flexibly than a decision tree would. ... 205

Table 25: A transcript from a session with 18 tutor turns, more than usual, again shows that IMP planned ahead to tutor effectively. However IMP too easily dismissed evidence of some gaps as mere slips, concluding by the end of the long session that the trainee did not need further help and ineffectually using four of its last six turns. ......................... 208

Table 26: A typical transcript from a training session where the participant asked questions. As often happened in the P condition also, IMP did not volunteer hints about differentiating friend and foe targets, but in the Q condition the trainee was able to ask questions during practice and address this knowledge gap. ........................................... 217

Table 27: An example of trainee questions unbalancing IMP’s learner model. At the end of the first practice the trainee used the QUI to ask all available questions about extension material. Following its learner model, IMP interpreted the advanced questions as evidence that the learner did not need help on any basic material. Later the trainee made
some mistakes, but few enough that IMP dismissed them as slips and did not respond effectively to address them.
LIST OF ABBREVIATIONS

**CFF** *Call For Fire*. The process by which forward observers coordinate with artillery and other units to describe enemy units and request attacks.

**CLQ** *Cognitive Load Questionnaire*. A specific instrument that estimates mental effort as a proxy for cognitive load as a whole.

**DBN** *Dynamic Bayesian Network*. A structure that models the probability of certain states based on the existence of other states, including states at earlier times.

**DVTE** *Deployable Virtual Training Environment*. A field-deployed, laptop-based training environment that lets Marines practice a wide variety of skills.

**FO** *Forward Observer*. A Marine who specializes in observing enemies from the front lines of a conventional battle.

**FOPCSIM** *Forward Observer Personal Computer SIMulator*. A program that runs on the DVTE and lets Marines practice CFF tasks alone, with computer teammates.

**IMP** *Inquiry-Modeling POMDP Adaptive Tutor*. An ITS developed to study this dissertation’s research hypotheses by supporting FOPCSIM training.

**ITS** *Intelligent Tutoring System*. A computer program that models learners in order to tutor or train them more adaptively.

**KSA** *Knowledge, Skill, or Attitude (elsewhere, “Ability”).* A learning objective, possibly itself composed of lower-level KSAs, that a learner should acquire.

**MDP** *Markov Decision Process*. A set of functions that model how actions affect the state of a system where each change depends only on the previous system state.
POMDP  Partially Observable Markov Decision Process. An MDP describing a system whose possible states are completely known but whose current state is not.

QUI  Question User Interface. The interface whereby IMP lets learners ask for help during training.
CHAPTER 1: INTRODUCTION

1.1 Intelligent Tutoring Systems for Effective and Efficient Instruction

Most learners have the potential to succeed. They can master required knowledge, skills, and attitudes (KSAs) and perform at high levels—but only under the right circumstances. When learners do not get the support they need, the consequences appear in decreased learning and lowered performance (Goldstein, Schwade, Briesch, & Syal, in press; Snow, 1992; Snow & Lohman, 1984; Talbert & Cronbach, 2002). One example is the work of Bloom (1984), who showed a dramatic difference between learners in a classroom setting and learners studying for the same amount of time with a personal tutor. The average tutored learner outperformed 98 percent of the classroom learners. If learning in a personalized, adaptive setting is so much more effective than learning in a group setting, then all learners deserve access to this educational tool that helps them reach their full potential.

At the same time, though, assigning a single teacher to a group of learners is more efficient than assigning many personal tutors. The average class size in US public secondary schools is 23.4 students (U. S. Department of Education, 2009). Theoretically, providing personal tutors in place of these classrooms would require 23.4 times the cost in teaching staff alone, but practically the proposition is simply impossible. Efficiency concerns also limit personal teaching and training at all other levels, such as job training for adults.

An intelligent tutoring system (ITS) is a tool with the potential to be both effective and efficient. Unlike non-adaptive computer learning tools, ITSs model users to
determine what help they need to learn better (see Figure 1). ITSs are effective because their instruction adapts to each individual learner. ITSs are also efficient because they can be deployed to a large group of students and support a single teacher in instructing all of them.

![Diagram of Computer-Based Learning and Intelligent Tutoring](image)

Figure 1: Intelligent tutoring systems are a subset of computer learning environments that “close the loop,” internally modeling users to determine the best way to help them learn.

However, the more effective ITSs currently require high development costs, as Section 2.1 describes. Intelligent tutors’ high cost to develop may help explain their disappointingly slow uptake in school settings. In addition, even the most effective ITSs still do not produce outcomes as good as human tutors can (e.g., Koedinger, Anderson, Hadley, & Mark, 1997). The difference in outcomes should be explained and addressed. Therefore, new research should focus on making ITSs more effective and making effective ITSs more efficient to produce.
On the other hand, improving ITS interactions in a naïve way might increase development costs, potentially by a large amount. Many ITSs already require great effort before they are ready for real students. Thus, in addition to increasing ITSs’ effectiveness, scientific improvements should also help lower the cost of developing effective ITSs. One possible method, a suite of related improvements to an internal model in the ITS, is explored in the present dissertation.

1.2 Inquiry Modeling

The heart of an ITS is its ability to adapt to an individual learner. Typically, an ITS will assess a learner’s performance, diagnose reasons for the assessment, and intervene as necessary. Assessment refers to judging raw observations, for example assigning a grade to a learner’s performance. Diagnosis refers to interpreting assessments in context to infer underlying causes for their appearance. Intervention includes deploying actions that are designed to address particular diagnoses, such as displaying hints or selecting the next instructional material to present. An ITS uses performance observations as evidence to update its user model with new estimates of each user’s states, traits, and misconceptions, and then its pedagogical module uses the diagnoses encoded in the user model to determine what material to present or help to offer.

Although ITSs’ diagnosis and adaptation abilities make them more effective, clear qualitative differences remain between learning with an ITS and learning from a human. Humans can draw on a wider range of inputs from moment to moment, are better able to estimate learners’ abilities and characteristics, and thus intervene more effectively.
One difference between existing ITSs and human tutors lies in the teaching interactions where tutees ask questions or request help. Many existing ITSs either do not allow such interactions at all, limit them severely, or allow them but do not effectively use the information they provide about a tutee. Improving help-seeking interactions could narrow the gap between ITS and human tutors’ efficacy. The present dissertation introduces inquiry modeling, a new framework for letting ITS users ask questions and leveraging them as an input source for the intelligent tutor learner model.

When students interact with a human tutor, they often ask many questions in the course of one session (Graesser & Person, 1994; Soler, 2002). The questions they ask indicate deficits in specific KSAs and therefore provide prime opportunities to teach about a particular topic without being too restrictive, removing a sense of control from the learner, or too permissive, allowing errors to propagate out of control (Merrill, Reiser, Ranney, & Trafton, 1992). By letting users ask questions, inquiry modeling can help an ITS detect an impasse earlier and supply more evidence to help determine the causes of impasses or errors when they do occur. In addition to information about learners’ KSA mastery, questions can also provide input about their personality and transitory states. Furthermore, ITSs that can answer questions for human instructors would lighten their workloads, and the learners’ questions could also be easily aggregated to enhance feedback to instructors.

The present dissertation work involves creating an inquiry modeling ITS for a complex instructional domain. Section 2.2 supports the inquiry modeling idea by discussing published work on how learner questions can improve tutoring and training.
Section 1.3, below, discusses an interesting learner model architecture that can support inquiry modeling.

1.3 POMDPs

Tutoring is well modeled as a problem of sequential decision-making under uncertainty. A model of the tutee is central to proactive tutoring, and many user models are possible to represent with a partially observable Markov decision process (POMDP). Solving a POMDP that models how a learner interacts with an ITS yields a corresponding policy. Then the ITS can use the POMDP model to interpret its assessments into diagnoses, and follow the previously generated policy to direct its interventions. The present section further elaborates how a POMDP representation aligns with a wide range of ITS needs and introduces terminology that will help discuss POMDPs throughout the rest of the work. Since POMDPs have not been widely applied in the ITS domain, the section concludes with an argument to build the intuition that a POMDP user model will enable important advances in ITS efficiency and efficacy, in general, and in inquiry modeling ITSs, in particular.

1.3.1 POMDP Definition

A POMDP is a formalism for representing problems that involve planning under uncertainty (Kaelbling, Littman, & Cassandra, 1998). Once represented in this formalism, a POMDP solver can be used to iteratively compute a policy for optimizing a reward function over progressively longer time horizons. POMDPs have already proven useful in many non-ITS planning tasks such as controlling robots (Thrun et al., 2000), planning
medical treatments (Hauskrecht & Fraser, 2000), and interpreting spoken dialogue (Young et al., 2010) or video (Hoey & Little, 2007). These applications highlight the potential power of POMDPs for finding long-term optimal actions in uncertain situations.

A particular POMDP is defined by a tuple \( \langle S, A, T, R, \Omega, O \rangle \) comprising a set of hidden states \( S \), a set of actions \( A \), a set of transition probabilities \( T \) that define how actions change system states, a set of rewards \( R \) that assign values for reaching states or accomplishing actions, a set of observations \( \Omega \), and a set of emission probabilities \( O \) that define which observations appear when the system is in a given state (Kaelbling et al., 1998). Figure 2, and the rest of this subsection, briefly describe how each of the POMDP components work together to model a tutoring problem.

\[
\begin{align*}
a. \text{ Initialization} & \quad \text{b. Tutoring Cycle} \\
\text{Stationary during user interaction} & \quad \text{Stationary} \\
\text{Time } 0 & \quad \text{Time } t \quad \text{Time } t + 1
\end{align*}
\]

![Figure 2: The POMDP representation aligns well with traditional ITS tutoring tasks and process flow (compare with Figure 1). After the ITS is constructed in advance (a), each interaction with a user updates the tutor’s user model (belief) and suggests the next action the system should take (b).](image-url)
The hidden states in $S$ can be thought of as statements about reality that the ITS cannot directly observe. The states are discrete and there are a finite number of them, all known in advance. At any given time, exactly one statement is true. The purpose of the POMDP is to describe how the states change, so a POMDP solver can estimate which state applies at the moment and propose plans to improve the state over time. To construct an example: a simple ITS might aim to teach users two topics, electronic troubleshooting skills and the layout of a specific printed circuit board (PCB). Then a corresponding POMDP might contain four hidden states, $S = \{s_0 := \text{user understands troubleshooting only}, s_1 := \text{user understands the PCB only}, s_2 := \text{user understands both}, s_3 := \text{user understands neither}\}.$

At any given time, one state $s \in S$ correctly describes the system. In general, though, the true state of the system is hidden from the ITS. Therefore, the ITS must maintain a probability mass function (PMF) over the states in $S$. The current probability distribution estimate is called the belief. When $S$ describes a user’s knowledge or mental states, the ITS’s belief acts as an up-to-date user model. To continue the example above, at a certain moment the ITS’s belief might assign the probabilities $P(s = s_0) = 0.25$, $P(s = s_1) = 0.15$, $P(s = s_2) = 0.10$, $P(s = s_3) = 0.50$. An initial value for the belief is part of the problem definition. The belief can only change based on actions the ITS takes and new information it observes.

The elements of $A$, the actions, define all the ways the ITS can change the system state. For each state and action, a matrix $T$ of transitions defines the possible outcomes in terms of a new system state. Example actions might be $A = \{a_0 := \text{administer a} \ldots \}$
knowledge test, \( a_1 := \text{tutor some troubleshooting skill}, a_2 := \text{teach about some PCB component} \). Before the ITS takes some action, the system can be in any of four states.

There are three possible actions, and each of them has some probability (possibly zero) of moving the system to another of the four states. For this example, then, \( T \) would be a four by three by four matrix. Each matrix element is a real value representing a probability of the system ending in a given state after the action. For example, if a learner actually understood neither troubleshooting nor the specific PCB layout, then teaching about some PCB component might be assigned a probability \( P = 0.10 \) of moving the system to the “understands PCB” state, \( P = 0.90 \) of staying in the “neither” state, and \( P = 0.0 \) of moving into either of the other two states. The very simple transition matrix in the example given already requires \( 4 \times 3 \times 4 = 48 \) parameters whose values must be learned or set by a subject matter expert. Of course, various sparse or functional representations of the transition matrix might make it more manageable in practical use.

After each action, the ITS may observe new information such as a user’s action in a simulator or response to a question. In discrete-event systems such as those discussed in the present work, each observation represents a change in the system rather than the passage of equal time quanta. Practically speaking, the passage of specific time intervals can still be prearranged as system events triggering observations. The set \( \Omega \) defines the possible observations the ITS can make, and it makes exactly one observation from \( \Omega \) at each opportunity. The observation emission matrix \( O \), in turn, defines the probability of each observation appearing based on the actual state of the system and the last action the ITS took. The example ITS might be constructed to expect the observations
\( \Omega = \{ o_0 := \text{user passed a knowledge test, } o_1 := \text{user failed on troubleshooting, } o_2 := \text{user failed on PCB layout, } o_3 := \text{user failed on both, } o_4 := \text{no assessment} \}. \) Then the dimensions of \( O \) would be four by three by five. The entries in \( O \) would be interpreted as probability statements such as, \textit{if the user knows about general troubleshooting but not this PCB, and the last action was to administer a knowledge test, then the probability of observing a pass is 0.10.} \textit{Non-zero observation emission probabilities are likely to appear often in tutoring problems because even learners who know material can slip, while learners who do not know the material can make lucky guesses.} Again, all 60 real elements of \( O \) must be learned or set by subject matter experts.

The final piece of the POMDP problem representation is the set of rewards \( R \) for each state, action, and observation. In the present work, rewards always vary only by state and never by action or observation, so rewards based on actions or observations will not be discussed further. On every action-observation cycle, the ITS garners a reward based on the actual state of the system. For instance, leaving a user in the most ignorant state would probably yield no reward, whereas moving the user into the most knowledgeable state would give the ITS a high reward. Rewards can be balanced to favor quick gains or long-term learning.

Solving a POMDP means selecting a policy \( \pi \) that tells the ITS what action to carry out based on the current belief. For a given belief, different policies will move the system through a different sequence of states, and it is the goal of the solver to select a policy that will maximize the reward. Planning ahead has a place in POMDP solutions, meaning that the best policy is not always the greedy one. In the PCB tutoring example, a
good policy will probably administer one or more knowledge tests when the ITS does not have a clear belief about what the user needs help to learn. Administering a test will not improve the user’s knowledge, so an immediate reward is sacrificed, but the test will let the ITS choose more effective actions on later opportunities and maximize the eventual reward. In the present work, policies are found offline, that is, solutions are fixed before use and do not change as the ITS interacts with users.

In conclusion, the generic POMDP representation of a problem aligns well with the processes and workflow many ITSs follow in modeling a user, assessing and diagnosing the user’s needs, and choosing adaptive interventions. The similarity supports the intuition that a POMDP may effectively control an ITS, while the important differences represent interesting directions for future research into their consequences. This section also introduced terminology that will appear throughout the present work.

### 1.3.2 POMDPs in ITSs

The POMDP representation introduced in the previous section lends itself to modeling many aspects of an intelligent tutoring system. If POMDP states are taken to model mental states of a particular user, then POMDP actions and transition functions naturally model the effects of ITS interventions, bringing in the ITS’s pedagogical module. Likewise, POMDP observations and emission functions do the work of ITS assessment and diagnosis modules. Thus, a POMDP user model has the potential to integrate with more aspects of the ITS than many conventional user models, such as those that merely classify users. The greater reach of the POMDP throughout the ITS adds to
the model’s future potential to build more intelligent interactions more efficiently with machine learning, although that potential will not be the subject of the present dissertation work.

The primary strength of a POMDP over other problem representations is its ability to plan a sequence of actions in the face of an uncertain situation (Kaelbling et al., 1998). Being able to model uncertainty, and work effectively despite it, makes POMDPs well suited for problems in the ITS field. Tutoring in general involves uncertainty because of the disconnect between the ground truth of a learner’s actual internal mental processes and the observable behaviors that the tutor must use to infer them. Learners’ mental processes and plans are often vague and uncertain (Chi, Feltovich, & Glaser, 1981). Learners who need help can still make lucky guesses, while learners who know material well can make mistakes (Norman, 1981). Understanding is fluid and changes often. It is difficult to exhaustively check all possible mistakes (VanLehn & Niu, 2001), and the effect of any particular intervention on a learner is rarely certain (Snow & Lohman, 1984). These sources of uncertainty during tutoring suggest that ITS assessments should not simply trigger black-and-white decisions about what the user does or does not know. Instead, optimal ITSs should employ diagnoses that are probabilistic and that influence multiple hidden states.

Despite the uncertainty inherent in teaching and training, human instructors and ITSs can still improve learning by successfully planning ahead. For example, a tutor might ask questions to confirm which underlying misconceptions caused an error, rather than simply correcting the most likely one (Graesser & Person, 1994). When a trainee
makes a mistake during practice, a trainer might decide to delay feedback to avoid overload (Hattie & Timperley, 2007). Even encouraging a frustrated learner is an example of planning ahead because it takes time away from immediate instruction to make later instruction more effective (Craig, Graesser, Sullins, & Gholson, 2004). These examples suggest planning ahead will improve efficacy if any conditions exist to make the simplistic, greedy choice ineffective. Therefore, a user model that handles uncertainty, as does a POMDP, would likely drive effective tutoring.

Alternative solutions without a POMDP might include a series of rules or heuristics for handling uncertain situations. Dynamic state estimation tools such as DBNs can detect uncertain situations but do not provide a method for planning a series of actions to handle them. Classical planners such as hierarchical task networks typically do not address questions of uncertainty (Poupart, 2005). Some contingent planners, however, enhance classical planning to compensate for possibly noisy observations and actions (e.g., Iocchi, Lukasiewicz, Nardi, & Rosati, 2009; Majercik & Littman, 2003; Onder & Pollack, 1999; Saad, 2009). Although contingent planners have abilities and advantages similar to POMDPs’, requiring them to handle uncertainty can lead to decreased efficacy and efficiency in solving real-world problems (e.g., Majercik & Littman, 2003; Ontañon, Mishra, Sugandh, & Ram, 2010). The POMDP architecture provides a framework to make sophisticated and exact plans based on many different inputs, on history, and on current hidden state estimates. POMDPs can interact with learners intelligently and increase their learning.
1.4 Contributions and Research Hypotheses

The present research into inquiry modeling and POMDP ITSs makes several contributions to the science. First, six general research questions are introduced in diverse topics ranging from behavioral science to machine learning. Next, the following subsections describe in greater detail four specific research hypotheses that focus the work.

1.4.1 Contributions of this Dissertation

The scientific contributions comprising this dissertation work are as follows.

1. *Inquiry modeling*: research conducted during this dissertation explores the practicality and impact of allowing users to ask questions of an ITS.
   a. Is it possible to let users ask questions of an ITS, and what development considerations does the ability require?
   b. What kinds of information can an ITS expect to glean from user questions?
   c. Does allowing users to ask questions of an ITS change or improve user acceptance, affect, or learning?

2. *POMDPs in ITSs*: changes introduced in this dissertation enable the application of POMDPs to intelligent tutoring system problems and explore their impact on ITS performance.
   a. What changes to the standard POMDP representation are needed to model the large-scale problems that ITSs face in the real world?
b. How will POMDPs’ ability to plan change or improve ITS behavior?

c. Are POMDPs able to effectively drive ITSs?

The capstone technological product of the present dissertation is an ITS that accomplishes training with inquiry modeling and a POMDP model. The process of designing, developing, and evaluating the ITS contributed to basic knowledge of these scientific questions. These contributions included an initial survey of ITS practitioners, a formative study with human participants, several related formative studies with simulated students, and a summative study testing the system with human participants in a full experimental design.

The present dissertation’s contributions to computer science and intelligent tutoring systems work together to support four research hypotheses.

1.4.2 Inquiry Modeling Hypothesis

The inquiry modeling hypothesis states that user questions contain information about users that could improve instruction if known, and that some of this information could not be inferred by observing performance alone. Such information potentially includes mastery of knowledge or skills, existence of specific misconceptions that interfere with mastery, and the current values of affective or other cognitive traits and states that can affect learning.

Besides the variety of information that might or might not be tested for, experimental circumstances also may change the amount of information that is available
through questions and through observing performance. For example, more structured
learning environments may allow close observation of performance, while more open-
ended environments may preclude inferring detailed information from performance
(Kodaganallur, Weitz, & Rosenthal, 2006). Likewise, different conditions may make it
easier or harder to ask questions (Harper, Etkina, & Lin, 2003; Marbach-Ad & Sokolove,
2000), or make the questions more or less useful (Folsom-Kovarik, Schatz, Sukthankar,
& Nicholson, 2010).

1.4.3 Scaling Hypothesis

The scaling hypothesis refers to the expectation that POMDPs can be effectively
applied to the ITS problem. Construction of new POMDP representations that scale to
handle problems of the large size that ITSs confront without sacrificing performance
would constitute evidence in support of the scaling hypothesis.

1.4.4 Planning Hypothesis

According to the planning hypothesis, the planning ability associated with
POMDP user models will effectively support question-asking in ITSs by updating user
models with the extra information that questions add and then adapting instruction based
on these data. This hypothesis implies that the ITS actions POMDPs select will be more
helpful than comparable overlay models with greedy action selection.
1.4.5 Summative Hypothesis

According to the *summative hypothesis*, an ITS that lets users ask questions and updates a POMDP user model with information from those questions will improve learning. The summative hypothesis represents a high standard, because many factors contribute to students’ learning. The summative hypothesis is not simply a combination of the other three hypotheses. First, the inquiry modeling hypothesis or the planning hypothesis may be true but produce only a small effect on learning under the conditions studied. It may even be the case that something as uninteresting as a clumsy interface for asking questions leads to a negative effect on learning. Second, the summative hypothesis may obtain through other routes than the inquiry modeling and planning hypotheses, such as by increasing users’ engagement or encouraging more active and constructive learning.

1.4.6 Published Works

This dissertation contains work previously published in the following papers:


This dissertation also includes work that is currently in preparation or in press in the following publications:


CHAPTER 2: SURVEY OF RELATED WORK

This chapter briefly reviews published research in several fields that relate to the present dissertation work. There are three sections in the chapter. Section 2.1 begins with introductory material describing a representative sample of existing intelligent tutoring systems. It organizes ITSs according to how they model their users, and analyzes them in terms of their reported efficacy and, where possible, their efficiency. The archived literature associates certain user models with greater efficacy than others, but to date none has matched the exceptional outcomes of a human tutor. Section 2.2 discusses research that suggests how inquiry modeling can improve the limits of ITS efficacy. Synthesizing important results from ITS and non-ITS research, this section paints a picture of the design decisions an inquiry-modeling system needs to consider. Finally, Section 2.3 describes research related to applying POMDPs to modeling and planning problems. This research highlights related applications where POMDPs have been effective and efficient and suggests benefits that an ITS built around a POMDP would enjoy.

2.1 Intelligent Tutoring System User Models

Intelligent tutoring systems instruct learners in highly personalized ways. Various ITSs respond to learners’ needs by detecting mistakes and correcting them, customizing their teaching styles, or even changing what material they teach. To make this adaptation possible, most ITSs contain internal models that reflect what they “know” about the learner, the material, and how to teach.
In particular, the *user model* lets an ITS adapt to different learner; consequently, it is central to the personalization benefits an ITS can offer. The user model also represents a large fraction of the overall ITS’s planning and development cost—about one third of the time needed to build an entire tutoring system (Folsom-Kovarik, Schatz, & Nicholson, 2010). This section focuses on the distinct types of user models used in ITSs to date and how each contributes to the benefits and costs of an ITS. In general, the historical user models discussed can give an ITS greater instructional efficacy or allow greater ease of development, but not both. Therefore, this section of the literature review underlines an existing need for research that improves ITS user models.

2.1.1 Evaluating User Models

2.1.1.1 Learning Effect Sizes

The primary goal of an interactive learning environment such as an ITS is to facilitate learning (Hasselbring, 1986). Researchers commonly report ITSs’ teaching effectiveness with a standardized measure called the learning effect size. The effect size statistic describes the difference in pretest–posttest improvement between a control group and an experimental group. For example, an effect size of 1.0 means that the average improvement in the experimental group was one standard deviation higher than the average improvement in the control group. Effect sizes remove sensitivity to variations between populations, and as such they are more comparable across studies (Schulze, 2004).
Especially for well-studied methods, different experiments may demonstrate different effect sizes for the same approach. Although this section relates several studies of each model type to show a range of possible outcomes, the highest reported effect size is used for comparison. The highest effect size reflects each model’s potential performance under, arguably, the best evaluation conditions. Since any model is capable of producing a small effect size if conditions are unfavorable, representing models with their maximum published effect size helps prevent possible bias against model types with more studies. This comparison approach does not penalize lower effect sizes, and it aligns with the thinking at the United States Department of Education, which accepts experiments as support for an educational method even if they show small or insignificant positive effect (U. S. Department of Education, 2008).

This literature review estimates ITSs’ benefits by surveying publications that compare an ITS’s effect on learning against a control condition with experimental or quasi-experimental designs on human, not simulated, participants in laboratory or applied settings. Twenty publications met these criteria for inclusion. Appropriate control conditions were not always the same in every publication; consequently, this section describes the control condition when reporting effect sizes.

2.1.1.2 Effect on Learning Is More Useful than Model Accuracy

Many factors contribute to the learning gain an ITS can realize—from the range of interventions at its command (e.g., Koedinger & Aleven, 2007; Wang, Johnson, Rizzo, Shaw, & Mayer, 2005) to the conduciveness of the learning environment outside the tutor
(Kirkpatrick & Kirkpatrick, 2006, pp. 23-25). This review makes the simplifying assumption that the practitioners creating a new ITS will improve these other factors to the greatest extent possible. The present survey only discusses the choice of a user model for the new system, assuming that the successes of each ITS as a whole can represent a proxy by which to judge its user model. Of course, in reality the causation is far less direct.

Despite the limitations, comparing ITSs’ learning effects is more useful than using other measures that evaluate models in isolation, outside the context of a working ITS. Some formative studies of user models compare their performance based only on the accuracy of the models’ predictions (e.g., Dam & Kaufmann, 2008; Hu, Xu, Huang, & Leung, 2009). Such accuracy measurements are useful for delivering statements of the type: *predictions using the new method are ten percent more accurate than the canonical method*. However, when evaluating user models for a new ITS, practitioners should recognize two problems with comparing models on their accuracy alone.

First, using a single number to measure accuracy can oversimplify the tradeoffs in model performance (Kubat, Holte, & Matwin, 1998). There are ways to address this objection, such as supplementing direct model comparisons with additional accuracy measures such as ROC curves, precision and recall, and goodness of fit (Yudelson, Medvedeva, & Crowley, 2008). However, even these more sophisticated measures of accuracy do not address the second important question: how does higher accuracy in a model divorced from a teaching system actually relate to the performance of a real interactive learning system *in situ*?
The relationship between modeling accuracy and improved learning rates is different for each ITS. If a new model is ten percent more accurate, that may have a large effect or a small one (Koren, 2008). In fact, it is possible to improve model accuracy but decrease teaching performance—for example, a new learner model might gain accuracy for almost every prediction but lose accuracy in predicting those few interactions that are crucial to react to a teachable moment. Because there is a disconnect between model accuracy and ITS performance, higher accuracy does not necessarily mean that a model can support good learning. For both ITS researchers and practitioners, helping students learn more is the ultimate goal, and learning effect size is the measure that best captures that goal.

2.1.1.3 Development Cost and Return on Investment

In an ideal world, the interactive learning system with the largest effect on learning would be the best. However, every system deployed in a real learning environment also has an associated cost. Much of this cost is the effort developers and domain experts expend to produce the system. As detailed in the rest of this section, creating a user model for an ITS may require up to the equivalent of five people working full-time for a year (Koedinger, Aleven, Heffernan, McLaren, & Hockenberry, 2004). On the other hand, the simplest models can be ready in just days (Blessing, Gilbert, Ourada, & Ritter, 2009; Folsom-Kovarik, Schatz, & Nicholson, 2010). There is a clear need to consider costs as well as benefits in planning the models for a new ITS.
User model costs are reported as ratios of development time in person-hours to learner interaction time in hours per individual. The ratio format makes cost figures more comparable across more or less complex ITSs. Twelve publications met the inclusion criteria of estimating both learner interaction time and development time of a user model, documenting 22 ITS development processes. Cost and benefit figures in this survey do not always refer to the same ITS, but are related by their common learner model type.

2.1.1.4 Abstraction

Systems that model learners with different levels of abstraction have different focuses and abilities. For example, an ITS that gives very specific hints about a learner’s errors will need a detailed model of learner cognition to diagnose each mistake. On the other hand, an ITS that reacts to errors with simple reteaching could maintain a more abstract model. At the highest level of abstraction, a user model might maintain a single score to represent mastery of the material.

Model abstraction forms a continuum from detailed to abstract, and from brief to long time scales (see J. R. Anderson, 2002; Newell, 1990). For the purposes of this review, ITSs model learner abilities at subtask, task, or skill abstraction levels. Subtasks are the most detailed and they represent small or atomic actions specific to a domain, like carrying the one in an addition task. Tasks usually represent groupings of subtasks into a level that would make sense to present as a single problem. Finally, skills and knowledge represent broad categories of competence. Models of sub-second cognitive events would be even more detailed than the subtask abstraction level, while models of month-scale,
social-level interactions would be even more abstract than the skill level. These extremes of detail and abstraction do not appear here because no existing ITSs represent events at these levels.

2.1.2 Survey of Models

The level of detail with which an ITS models its learners determines both how much the ITS can be said to know about a learner and the amount and kinds of work required to build the model. Although the literature to date provides insufficient examples for a completely reliable meta-analysis, model abstraction may help account for the intuitive trend that this survey objectively supports: abstract learner models are possible to develop with less effort, but detailed learner models are capable of stronger effects on learning.

Rather than a chronological survey, the subsections of Section 2.1.2 describe model types from the most detailed to the most abstract. ITSs that need to model all subtasks in a domain can use production rule systems, buggy or perturbation models, or constraint-based systems. If only a few subtasks need to be modeled, then example-tracing systems and overlay models might also be appropriate. At the task abstraction level, appropriate model types include example-tracing, constraint-based models, Bayesian and other classifying models, and overlays. Finally, at the levels of skill estimation or higher abstraction, Bayesian networks, classifiers, and overlay models are appropriate.
2.1.2.1 Production Rules

The most detailed models currently used in ITSs are the production rule systems that employ a model tracing algorithm. Model-tracing tutors based on ACT, ACT*, or ACT-R models are also called cognitive tutors because they model students’ cognitive steps at the very granular subtask level, such as selecting a theorem to use in a geometry proof (J. R. Anderson, 1993). The learner model works by encoding each of these subtasks as a single production rule that matches certain inputs and transforms them into a specific output. The model-tracing algorithm tries possible production-rule combinations to find chains that reproduce the behaviors the ITS actually observed in the learner. Several early production-rule systems based on cognitive theories other than ACT-R operated on a similar subtask level (Neches, Langley, & Klahr, 1987), and there are also modern ITSs that trace production rules at the task level without reference to ACT-R (Callaway et al., 2007; B. G. Johnson, Phillips, & Chase, 2009). However, most cost and benefit results described in the published literature refer to the more prevalent ACT-R ITSs.

Anderson’s original model-tracing systems needed between 100 and 1000 hours of development for each hour of instruction, which he presented as comparable to the time “traditionally” required to build computer-aided instruction systems of the time (J. R. Anderson, 1993, p. 254). The firmest cost figure comes from a model-tracing algebra tutor that took approximately 10,000 hours to develop and provided 50 hours of instruction, or 200 development hours for each hour of instruction (Koedinger et al., 2004).
More recently, simple model-tracing tutors have been built with the new Cognitive Tutor Authoring Tools (CTAT), a set of tools designed to speed the authoring process. A formative study, where four graduate students built a model with just six rules, showed that using CTAT sped up the development process by 40 percent (Aleven, McLaren, Sewall, & Koedinger, 2006). If this result can generalize to larger projects, future model-tracing tutors might require less than 100 hours of development time for each hour of instruction.

Early model-tracing tutors demonstrated success in several settings. A high-school geometry tutor produced improvements of “more than one standard deviation” compared to classroom study (J. R. Anderson, Corbett, Koedinger, & Pelletier, 1995, p. 183). A tutor for teaching college students the LISP programming language produced an effect size of 0.75 compared to working problems without hints, and also cut instruction time by two thirds (Corbett, 2001; Corbett & Anderson, 2001).

Model tracing formed the basis for one of the milestone studies on ITS effectiveness (Koedinger et al., 1997). In conjunction with an overhauled curriculum, the PUMP Algebra Tutor (PAT) taught algebra skills to a large sample of ninth-grade students in urban schools. Compared to students who received all their math instruction in a classroom, students who used PAT for 25 out of 180 class periods displayed improved learning with an overall effect size of 0.7 and a 1.2 effect size on tests that covered the same material as the tutor. This improvement did not come at the expense of other math topics, since the students also displayed an improvement on standardized math tests with an effect size of 0.3.
As shown in this section, even the cheapest production-rule tutors require a time investment at the high end of all model types. Some researchers who use model-tracing systems argue that implementing a specific cognitive theory is worth the added cost (Neches et al., 1987), and the empirical study described above does show that detailed cognitive tutors can produce learning increases among the best of any ITS.

2.1.2.2 General Perturbation Models

A broad class of learner models called *perturbation models* or *buggy models* try to describe all the incorrect knowledge the learner may have. Incorrect knowledge, variously called misconceptions, mal-rules, or bugs, represents persistent errors in thinking that the ITS should know how to correct. The model-tracing ITSs described above could be considered relatives of buggy models because their production rules include incorrect rules for the purpose of diagnosing students’ misconceptions (VanLehn, 1988).

Buggy models usually specify misconceptions at the subtask level, and as with cognitive tutors, the misconceptions are specific to a domain but grounded in theories of cognition. The link from domain-specific misconceptions back to cognitive theory allowed systems like DEBUGGY (Brown & VanLehn, 1980) and IDEBUGGY (Burton, 1982) to automatically generate plausible bugs based on their “repair theory” of learning. Alternatively, other systems such as Proust (W. L. Johnson, 1990) depended on lists of bugs exhaustively enumerated by a domain expert rather than generated by a cognitive theory.
Like cognitive models, buggy models can entail a high cost because of the need to list large libraries of bugs and their optimal remediation strategies. Worse, the job may never be complete because different learner populations may display different misconceptions (Payne & Squibb, 1990). However, no published accounts give specific figures for system development costs.

Over time, buggy models have somewhat underperformed production-rule models in improving learning. One buggy-model ITS, called Smithtown, let students explore and manipulate an artificial economy (Shute & Glaser, 1990). Students who spent five hours with the ITS did as well on a test of economic principles as a control group of students who spent eleven hours studying in a classroom. However, in the same time period PIXIE, a buggy-model ITS that taught introductory algebra, found that remediating specific bugs was not more effective than simply re-teaching the material (Sleeman, Ward, Kelly, Martinak, & Moore, 1991). This negative result was widely cited, and it seemed to cast a pall over research into buggy learner models (e.g., Self, 1990). However, a simple explanation may be that re-teaching helped students solve superficially similar problems, while leaving untouched the underlying misconceptions (Chi, 1996).

More recently, a learning system called Adaptive Content with Evidence-based Diagnosis (ACED) used a buggy model for error feedback while teaching algebra to high school students (Shute, Hansen, & Almond, 2008). ACED also used a Bayesian model (see Section 2.1.2.5) for problem selection, but a careful statistical analysis showed that only the buggy model component contributed to significant learning improvement. ACED did show a positive effect size of 0.38, but only in comparison to no intervention.
at all. The main focus of the ACED project was not on improvement in learning, but on improving assessment without damaging students’ learning.

Finally, two dialog-based tutors, AutoTutor (Graesser et al., 2004; Graesser, Wiemer-Hastings, Wiemer-Hastings, & Kreuz, 1999) and Atlas (VanLehn et al., 2002), both estimate students’ knowledge with short lists of common misconceptions—on average, fewer than five (VanLehn et al., 2007). Although these small bug lists are not the focus of the ITS authors’ research, they play some role in controlling the ITSs. The current literature does not describe the development cost of such small buggy models. In various studies, learning with AutoTutor as compared to textbook reading improved learning with effect sizes of 0.50 (Person, Graesser, Bautista, Mathews, & the Tutoring Research Group, 2001), 0.31 (Graesser, Moreno, et al., 2003), 1.02 (Graesser, Jackson, et al., 2003), and up to 1.02 (VanLehn et al., 2007). However, under slightly varied conditions, improvements with both Atlas and AutoTutor were not significant (VanLehn et al., 2007). Furthermore, a small study comparing ITSs having the same internal model but different dialogue capabilities attributed much of the effects on learning to the dialogue interaction rather than the learner model (Rosé et al., 2001). Consequently, dialog-based tutors’ impressive effect sizes are less relevant for the present report about learner models.

In sum, detailed buggy models require significant effort to build a comprehensive library of misconceptions and targeted interventions. Furthermore, the published literature lacks examples of buggy-model ITSs that demonstrated a significant improvement in learning effectiveness. However, exceptions to these findings come from
dialogue-based ITSs that successfully employ very abstract buggy models containing only a few misconceptions.

2.1.2.3 Example Tracing

Like tutors with buggy models, example-tracing tutors can also respond to errors that learners make at the subtask level. However, example-tracing tutors do not maintain a general list of possible misconceptions. Instead, the system’s authors predefine specific incorrect responses for each question. Example-tracing tutors are less concerned with the overarching cognitive theory that underlies these mistakes. Since piecemeal examples take the place of the more generalizable production rules cognitive tutors use, example-tracing systems do not require content authors to articulate a cognitive model. This abstraction away from a detailed cognitive theory also gave example tracing systems their old name, pseudo-intelligent tutors or pseudotutors.

Example tracing models were created as a direct response to the high development cost of using the model-tracing approach (Koedinger et al., 2004). From their inception, development of example-tracing systems has been sped by a suite of authoring tools, the Cognitive Tutor Authoring Tools (CTAT) that are also used for creating model-tracing systems. In preliminary tests, domain experts needed an average of about 23 hours of design and development time to create one hour of instruction with CTAT (Koedinger et al., 2004). ASSISTment Builder, an outgrowth of CTAT, let novice content authors create new example-tracing math problems in only minutes. In two
studies, novices required approximately 30 (Heffernan et al., 2006) to 40 hours (Razzaq et al., 2008) of development time to create one hour of instruction.

In the future, it may be possible to speed the development process even further by creating examples based on records of real student mistakes, rather than *ex nihilo* (Harrer, McLaren, Walker, Bollen, & Sewall, 2006). This idea has not yet led to practical improvements in development effort, but if successful, it could both increase the efficiency of the authoring process and improve the coverage of examples in new systems.

Example-tracing tutors have been evaluated in the laboratory and in the real world, such as the ASSISTment tutor for teaching high-school math. Compared to pen-and-paper homework, the ASSISTment tutor produced a learning improvement of 0.61 standard deviations (Mendicino, Razzaq, & Heffernan, 2009). Hockenberry (2005) used an example-tracing tutor to teach logic puzzle strategies to college students, and produced an effect size of about 0.75 compared to pen-and-paper practice. Furthermore, building this tutor with CTAT required only 18 hours of development for one hour of tutoring time.

Example-tracing ITSs are more abstract than canonical cognitive tutors and buggy-model tutors, in that at their most detailed they model only a subset of the learners’ subtasks. Published experiences show it is possible to build an example-tracing tutor in the least time of any model type. However, example-tracing tutors at their best have only produced moderate learning gains.
2.1.2.4 Constraint-Based Modeling

Constraint-based modeling eschews extensive models of learner cognition and instead constructs libraries of domain-relevant constraints against which learners’ actions are compared (Ohlsson, 1994). Constraint-based models need not track historic performance or even specific user actions, but instead monitor the immediate problem state. As long as a learner never reaches a state that the model identifies as wrong, he or she may perform any action. This architecture allows constraint-based tutors to selectively abstract away subtask details and concentrate on tasks, subsets of subtasks, or a combination thereof.

An early constraint-based tutor that taught SQL to graduate students had a cost ratio of 220:1 (Mitrovic & Ohlsson, 1999), which was approximately the same as the cost estimate for creating a different model-tracing cognitive tutor. A later direct comparison of constraint-based and model-tracing tutors that taught the same material was even more favorable toward constraint-based models. Since the production rule structure was “somewhat more complex” to develop, the model-tracing tutor took slightly longer to develop, but four times as long when including the time needed to learn each architecture (Kodaganallur, Weitz, & Rosenthal, 2005, p. 141).

Several authoring tools exist to help develop constraint-based tutors more rapidly. For example, graduate students used the Constraint Authoring System (CAS) to build a constraint-based tutor for adding fractions; on average they completed their tutors in 31.3 hours (Suraweera, Mitrovic, & Martin, 2007). Using CAS, the best-performing novices could create each constraint with almost as little effort as expert authors needed to build
the individual constraints of a larger system (Mitrovic & Ohlsson, 1999). The Web-Enabled Tutor Authoring System (WETAS) similarly helped four graduate students work together to create a small spelling tutor, requiring 32 person-hours to complete (Martin, Mitrovic, & Suraweera, 2008). Unfortunately, the teaching times of the above-mentioned tutors were not reported.

The authoring tool ASPIRE was least limited of all and could generate most of the constraints in two preexisting, full-scale constraint-based tutors. ASPIRE automatically extrapolated constraints from a few example problems and solutions (Mitrovic et al., 2006). A later effort added frameworks for authoring common problem types more easily, which allowed ASPIRE to construct a functional medical imaging ITS with ten hours of content (Martin, Kirkbride, Mitrovic, Holland, & Zakharov, 2009). Promising work on reducing ITS development effort continues, but so far the amount of human effort authoring requires has not been reported.

Several well-studied ITSs have used constraint-based methods, giving a good overview of the typical learning effect they can achieve.

CIRCSIM-Tutor trained medical students in the workings of the vascular system and detected students’ misconceptions with a constraint-based model. The constraint-based model, combined with several other modeling components and reconciled with simple precedence rules, drove microadaptation in the form of selecting natural-language dialogue moves (Zhou & Evens, 1999). CIRCSIM-Tutor produced significant learning (Michael, Rovick, Glass, Zhou, & Evens, 2003), but the improvement over reading a text about the same material was not statistically significant (Evens, 2003).
A tutor with a constraint-based model that taught SQL to college students in a single, two-hour session produced an effect size of 0.63 compared to an ablated version using no adaptation or feedback except for displaying the correct answer (Suraweera & Mitrovic, 2004). Another constraint-based tutor in the same domain had previously produced an effect size of 0.75, but in that formative study students in the control condition were simply those who did not volunteer to use the tutor for an extra session, so selection bias may have confounded that result (Mitrovic & Ohlsson, 1999).

Although the first version of the Andes physics tutor controlled microadaptation with production rules and macroadaptation with a Bayesian network, a second Andes version replaced both learner models—in part to improve return on investment in its particular environment—eliminating macroadaptation and accomplishing microadaptation more simply with constraints (VanLehn et al., 2005). Evaluated annually over four years, Andes with a constraint-based learner model yielded an overall effect size of 0.61 compared to working practice problems on pen and paper. The learning gain was especially large, 0.70 and 1.21, in the two areas where the Andes material most closely aligned with the course test material (VanLehn et al., 2005). This effect size is very close to the result reported for the model-tracing PAT tutor described above.

Finally, a constraint-based ITS taught a small group of students both domain material—constructing UML diagrams—and collaboration activities. Collaboration skills were introduced to all students with a brief talk and a paper handout. When the ITS additionally provided immediate feedback on collaboration activities, domain learning remained constant and knowledge about collaboration tactics improved by 1.3 standard
deviations, compared to students using the ITS with no collaboration feedback (Baghaei & Mitrovic, 2007).

The results described above show that constraint-based models have achieved better development costs than comparable production-rule systems. However, the results do not yet clearly determine the absolute size of the difference. On the other hand, constraint-based tutors have clearly been shown to be capable of producing very high learning gains.

2.1.2.5 Bayesian Networks and Other Classifiers

A Bayesian network consists of a collection of random variables, some of which have known values, such as how well a learner is performing on a particular test question. The unknown variables, such as how well that learner understands the underlying information, are inferred from neighboring conditional probabilities. The model includes dependence relationships between known and unknown variables that make its predictions reasonable (Charniak, 1991). These relationships are key—they allow one piece of new data to refine several interrelated estimates.

Like the other classifiers described in this section, Bayesian networks often model students at the task or skill abstraction levels, because Bayesian networks become more difficult to adjust as their size increases. A Bayesian network large enough to differentiate the hundreds of subtask-level misconceptions that some bug libraries can detect would be difficult to initialize, and its estimates would become highly suspect (Ott, Imoto, & Miyano, 2004).
Wayang Tutor used a Bayesian network with the relationships determined by machine learning. The network could interpret data such as the time needed to solve a problem, number of hints requested, and correctness of answers to drive problem and hint selection, without tracing every step of problem solving (Arroyo, Woolf, & Beal, 2006). One study showed that the tutor helped students learn, but the control and experimental groups were dissimilar and no effect sizes could be given (Beal, Walles, Arroyo, & Woolf, 2007). Another study found Wayang Tutor was as effective as small-group study with a human tutor (Beal, Shaw, & Birch, 2007), while comparing Wayang Tutor to study in a classroom or with non-interactive websites yielded an effect size of 0.39 (Arroyo, Woolf, Royer, Tai, & English, 2010).

Dynamic Bayesian networks (DBNs) are Bayesian networks that can account for the way data change over time. The Prime Climb educational game for teaching number factorization to children used a network similar to a DBN to model students’ affect and adapt its hints accordingly. The game showed a gain in learning with an effect size of 0.7 as compared to students who played the same game with no hints at all. A “modified roll-up” method made inference tractable and made the design of the dynamic network only slightly more complicated than the design of a static one. (Conati & Zhou, 2004)

Ecolab used a Bayesian network to estimate students’ ability level on each skill in the domain (Luckin & du Boulay, 1999). With the student model selecting material and support methods, learning gains in a preliminary study were 1.46 standard deviations better than in a self-directed learning condition. However, this preliminary result with a small sample size is difficult to compare to other studies because student control of
learning is known to result in much poorer performance than outside control, especially for novices and children (Kelly, 2008; Pressley & Ghatala, 1990). The learner model from Ecolab was later expanded in HOMEWORK (Luckin et al., 2006), which has not yet been evaluated in a statistical study.

Other manually constructed and machine-learning classifiers can also play the role of a student model. Like Bayesian networks, these methods come to ITSs from the computer science field, and some model cognitive science constructs at a very abstract level. Classifiers in ITS student models typically examine data from individual students and sort them into one group or another. The groups can represent competence at a skill or task, or they can represent even broader classifications, like concrete versus abstract thinkers. Examples of classifiers that have been used as student models include decision trees (e.g., Cha et al., 2006; S. W. McQuiggan, Mott, & Lester, 2008), neural networks (e.g., Castellano, Mastronardi, Di Giuseppe, & Dicensi, 2007), case-based reasoning libraries (e.g., Kass, Burke, Blevis, & Williamson, 1994; Reyes & Sison, 2002), and ensemble methods (e.g., Hatzilygeroudis & Prentzas, 2004; Lee, 2007).

The various classifiers may be more or less difficult to build. Bayesian models in particular can require care in defining their internal relationships. Often experts must estimate all the relationships; however, it is also possible to define the relationships by analyzing experimental data or to use preliminary relationships and refine the model as new data become available (Conati & Maclaren, 2005). Published reports typically do not detail the development effort needed to create classifiers. In part, this may reflect researchers’ belief that the various classifiers are “off-the-shelf” technology and their
implementation takes insignificant effort compared to other tasks such as knowledge elicitation. Some reports about building classifiers, while not specific about costs, show little effort may be required. For instance, several military tactics tutors, created with the Internet ITS Authoring Tool, used case libraries to evaluate student actions in simulations, which made developing the ITSs take “a small fraction of the time normally required” (Stottler, Fu, Ramachandran, & Vinkavich, 2001, p. 1). The Cognitive Model SDK is an authoring tool for manually developing hierarchical rules whose predicate sets function similarly to decision trees. This architecture let undergraduate novices develop the model for a fraction addition tutor in about a quarter of the time novices in a separate study needed to develop a constraint-based model in CAS for a similar tutor (Blessing et al., 2009).

Although this survey may seem to have painted the Bayesian and other classifiers with a broad brush, they all model students at a high level of abstraction and, in at least some practical situations, these tools are interchangeable in their performance (S. W. McQuiggan et al., 2008; Walonoski & Heffernan, 2006). No classifier model has been able to produce more than moderate learning gains. However, they are possible to build with low development costs.

2.1.2.6 Overlay Models

Many ITSs, especially early examples such as Scholar (Carbonell, 1970), PLATO West (Burton & Brown, 1976), and Wusor II (Carr, 1977, p. 66), modeled learners’ knowledge with an overlay. Overlay models are the most abstract models modern ITSs
use. They ignore details of how students learn, and instead track what students have learned in a simple way. Similar to a checklist, an overlay model specifies the knowledge and skills the ITS must impart. The ITS models each learner’s knowledge as an overlay, or subset, of the ideal knowledge set, with a perfect expert having the equivalent of a checkmark next to each skill. A student using the program for the first time is typically assumed to have a small or empty subset of the experts’ skills, and successful or unsuccessful performance in the tutor grows or shrinks the size of the overlay until it includes all the skills of an expert. Overlay models’ high abstraction does not lead to flexibility in learner interactions. On the contrary, overlays tend to force learners into specific answers and discount learner knowledge that falls outside the ITS’s model of expert knowledge (Burton & Brown, 1976).

Developing overlay models requires expert knowledge of the domain in order to specify topic definitions, prerequisites, and ordering that tell how to interpret the overlay and select the next topic or problem to present. After knowledge elicitation about the domain, there are few technical challenges to building the model itself.

Sherlock (Lesgold, Lajoie, Bunzo, & Eggan, 1988), which taught electronics troubleshooting, implemented a rather intricate overlay model. Sherlock’s overlay was not simply binary but could estimate trainees’ abilities as being at one of four levels for each skill in the model, and the model allowed for learning and forgetting. Comparing trainees who worked for 20 hours with Sherlock against those who had 20 hours of on-the-job instruction gave an effect size of 1.02 (Shute, 1990).
Because of their high abstraction away from a cognitive theory, overlay models are often selected when designers are not focusing their efforts on the learner model, and indeed there is at least one documented experience to support the intuition that it is possible to develop overlays with little effort. However, overlay models’ best effects on learning performance are no better than moderate.

2.1.2.7 Relationships to POMDPs

ITS practitioners familiar with other model architectures may find differences when considering the POMDP models this dissertation discusses. Compared to rule-based cognitive tutors, POMDPs operate at a higher level of abstraction. They have no need to specify neurological events (but no ability to leverage them either). POMDPs are more closely related to knowledge tracing models. Knowledge-tracing models are overlay-like structures that augment some cognitive tutors (Corbett & Anderson, 1995) and can themselves be machine-learned Bayesian networks (e.g., Baker et al., 2010; Ritter et al., 2009). However, POMDPs may also model non-mastery states that affect learning.

Constraint-based ITSs are another example where POMDPs are more similar to an adjunct structure. POMDPs are related to constraint-based tutors’ long-term models (Mayo & Mitrovic, 2001) and affective models (Zakharov, Mitrovic, & Johnston, 2008) because both can make inferences about multiple states from a single assessment and can differentiate a misconception from a momentary slip. In a POMDP, the policy selects optimal actions according to Bayesian principles, but POMDP policies are typically
determined only once while the constraint-based long-term model is updated during student interaction.

Manually constructed and machine-learned classifiers can also play the role of a learner model. Classifiers in ITSs typically sort individuals into broad groups based on mastery or cognitive traits. POMDPs differ from methods that merely classify learners because they integrate actions and action planning in the same model. Also, neural networks and very complex decision trees, such as ones that can handle many uncertain situations, can be opaque—making decisions that are difficult to explain. In contrast POMDPs can report their beliefs and recommendations in ways that users who are not expert in Bayesian principles understand (Almond, Shute, Underwood, & Zapata-Rivera, 2009).

2.1.3 ITS User Modeling Summary

Table 1 summarizes the development costs and instructional effectiveness of ITS learner models based on a survey of the published literature. The chart describes the current state of the art in various learner model types, but it does not represent their absolute limits—only their performance in the field and the laboratory to date. The information in Table 1 appears again in Section 5.1, with updates to include new information gathered from ITS experts during the course of this dissertation work.
Table 1: ITS costs and outcomes by learner model type based on published literature review.

<table>
<thead>
<tr>
<th>Student model</th>
<th>Model detail</th>
<th>Lowest reported development to learning time ratio</th>
<th>Highest reported effect on learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production rules and model tracing</td>
<td>High: all subtasks</td>
<td>200:1</td>
<td>1.2, compared to classroom learning</td>
</tr>
<tr>
<td>Perturbation and buggy models</td>
<td>High: some or all subtasks</td>
<td>No reports</td>
<td>Not significant</td>
</tr>
<tr>
<td>Example tracing</td>
<td>Moderate: some subtasks, not all; sometimes tasks</td>
<td>18:1</td>
<td>0.75, compared to paper homework</td>
</tr>
<tr>
<td>Constraint-based models</td>
<td>Moderate: some or all subtasks or tasks, or a mix</td>
<td>220:1</td>
<td>1.3, compared to briefing and handout</td>
</tr>
<tr>
<td>Bayesian networks and other classifiers</td>
<td>Low: tasks or skills</td>
<td>No reports</td>
<td>0.7, compared to the learning task with no hints</td>
</tr>
<tr>
<td>Overlay models</td>
<td>Low: tasks or skills; or some subtasks</td>
<td>No reports</td>
<td>1.02, compared to on-the-job training</td>
</tr>
</tbody>
</table>

2.2 Help Seeking and Question Asking During Learning

Seeking help represents an important part of learning, and willingness to ask for help correlates with other adaptive cognitive and metacognitive strategies (Karabenick & Knapp, 1991). Conversely, not seeking help when it is needed can have disastrous consequences, even for very capable people (Sandoval & Lee, 2006).

When a learner asks a question of an expert human tutor—the pedagogical gold standard—that help-seeking act is actually the culmination of a process first described by Nelson-Le Gall (1981). The steps of the help-seeking process are quoted below:

1. Become aware of a need for help.

2. Decide to seek help.
3. Identify potential helper(s).

4. Use strategies to elicit help.

5. Evaluate the help-seeking episode.

Effectively negotiating the process of asking a question is a metacognitive skill at which untrained learners often perform poorly (e.g., Aleven & Koedinger, 2000; Walonoski & Heffernan, 2006). For example, the first step of the process could break down if the learner does not notice a misconception (VanLehn, Siler, Murray, Yamauchi, & Baggett, 2003), or one of the later steps could be blocked by hesitation to interrupt, belief that help will not be effective, or other social and affective factors (Newman, 1998). Breakdowns of the help-seeking process can cause learners to ask few or ineffective questions, even when working one-on-one with human tutors (Graesser & Person, 1994).

Although students may find it difficult to ask effective questions, engaging in successful help-seeking leads to better learning (e.g., Webb, Ing, Kersting, & Nemer, 2006). Fortunately, instructors can teach, or otherwise encourage, better question-asking (Harper et al., 2003; Marbach-Ad & Sokolove, 2000). In fact, at least one ITS directly teaches effective help use in many domains (Roll, Baker, Aleven, McLaren, & Koedinger, 2005). Future ITSs could build on this trend to encourage help-seeking even more than human tutors do, for example by mitigating the interpersonal factors that raise a barrier to asking questions.

The present section will focus on questions that are instrumental to learning, as opposed to questions that derive from in-character behavior during a simulation or are
otherwise inherent to the narrative of a scenario. The distinction is necessary because when a learner takes on a role within an instructional simulation that role may normally require question-asking behaviors. For example, a doctor in a simulated emergency room may ask questions to gather a medical history from a simulated patient (Domeshek, 2008) or isolate faults to troubleshoot a circuit (Brown, Burton, & de Kleer, 1982). Questions like these are designed to identify the current situation, not to learn new material. ITSs already commonly handle such in-character questions, in the same way that they process and respond to other behaviors they are programmed to train. The purpose of such questions differs from questions designed for help-seeking, and they might not be created through the Nelson-Le Gall help-seeking process discussed above. This dissertation work focuses on ways an ITS could allow questions that are instrumental to learning and make them as useful to the learner model as performance measures such as in-character questions already are.

The present section organizes previous related work according to a novel taxonomy of question support for computer-based learning environments. The taxonomy contains three orthogonal dimensions that ITS practitioners, in contrast to human tutors, trainers, and pedagogical experts, must consider:

1. *Input freedom* describes the constraints the ITS places on learners’ questions with its user interface.

2. *Model integration* describes how learners’ questions update the ITS’s internal model.
3. *Response characteristics* describe how the ITS answers learners’ questions.

2.2.1 *Input Freedom*

In order to leverage learner questions in an ITS, the system must first allow the learner to ask questions. The system interface can give learners freedom to ask whatever they want, or it can impose constraints on their questions. The learners’ freedom to choose their own questions has an impact on the information the questions can convey to the model, on the ease of implementation, and on the user experience.

Figure 3 lists some ways learners may request help in different ITSs. The interfaces are located along a continuum from *constrained* to *free*. As an ITS’s question interface moves higher in the freedom dimension, learners become more able to ask any question they please.
2.2.1.1 Effects on Questions’ Information Content

Freedom impacts how questions can update a learner model. With more constrained interfaces, questions convey less information about learners’ mental states. For example, in a common constrained interface, a trainee can click a hint button within an ITS, which may then use that act to infer that the trainee is unsure either about what to do next or about how to do it. However, from the hint request alone the system cannot conclude which difficulty the learner has or what obstacle is causing the difficulty. The ITS must instead consult the learner model to surmise a best estimate of what has gone wrong.

In contrast, if an ITS can accept a question chosen from among several options on a menu, then the learner’s question choice more precisely pinpoints the problem. Rather
than consulting the learner model to get more information about the help request, the ITS derives the information directly from the learner and, if properly equipped, it can actually update the learner model with the new information. In this way, the freer interface raises the upper bound on the amount of information the ITS can infer from a question.

The interface category with the highest degree of freedom shown in includes natural-language typed, handwritten, and spoken questions. Because such a free question interface does not limit what the learner can ask, the learner’s questions have the potential to contain even more information. Whether a particular ITS has sufficient capability to understand natural language, and its learner model has enough detail to take advantage of the expanded information, will determine whether the ITS is able to realize the potential of its freer interface and extract more information from each question.

For completeness, the input freedom scale includes a zero point to describe ITSs that offer hints based only on student performance or other factors in the learner model. Help in such an ITS must be entirely driven by the learner model, with no learner initiative, and as a consequence the learner model cannot glean new information from the help episode. Although giving help without waiting for a request can be important and is not uncommon in ITS interactions, this mode of interaction can hardly be termed a question and will not be discussed in the present section.

2.2.1.2 Effects on Learning

There are several ways that the freedom to ask questions might affect an ITS’s teaching or training effectiveness, and the freest question-asking interfaces may not
necessarily produce the optimal results. For instance, more constrained question-asking interfaces may put less strain on learners’ cognitive workload, and they may help teach learners more effective help-seeking behavior. Further, such interfaces can be more readily created by ITS developers.

Asking a question is a mental task that may compete for cognitive resources that a learner could otherwise use to accomplish schema acquisition and automation—i.e., learning (Aleven, McLaren, Roll, & Koedinger, 2006; Sweller, 1988). In particular, framing a question and expressing it in words require mental effort (Newman, 1998). More constrained interfaces, like clicking a single button or choosing a question from only a few options, remove the need for these intermediate steps and might help to lighten the cognitive load. Users may be unsure about the natural language understanding and domain knowledge capabilities of a computer system with a free interface, while a constrained interface can make the system’s abilities clearer (Brown et al., 1982).

In addition, the ability to suggest questions may also let more constrained interfaces support the first step of Nelson-Le Gall’s (1981) model, which is to identify the need to ask a question. For example, an ITS with a question menu could detect a misconception in the learner and populate its menu with suggestions that point out the contradiction (B. G. Johnson et al., 2009). This ability would be less natural, though not impossible, with completely free help request inputs.

Placing constraints on a learner’s help-seeking process also has the potential to add scaffolding (Clarebout & Elen, 2006) to help train a skill that is known to be difficult for many learners. For example, presenting a menu of questions could give an ITS the
opportunity to suggest deeper or more effective questions than the learner might have thought of alone. In this way a more constrained interface could act as a role model for how to ask good questions, as advocated by Newman (1998).

Finally, there may be practical issues associated with greater interface freedom in a simulated environment. One ITS that required trainees to type during a simulated infantry task found that the trainees could not both type and keep up with events in the simulator (Jensen et al., 2007). Allowing speech input instead of typing may address this issue in some environments. In environments that also limit trainees’ speech, the simulator may even need to pause or otherwise accommodate the distraction of entering a question.

As described in the present section, greater input freedom may come with higher costs both for implementers and for learners, while constraining inputs can give learners valuable scaffolding. These considerations may have led some ITS designers in the past to avoid free question interfaces. However, the previous section suggested that more free interfaces give more potential to infer data about the learner from each question, while imposing constraints lowers the upper bound on the amount of available information. Effectively integrating the most possible question information into the learner model may require more freedom to ask questions than traditional ITSs have previously supported.

2.2.2 Model Integration

The questions a learner asks a tutor contain a wealth of information about that learner. The information can be explicit in the meaning of the question, but it can also be
implicit in the timing, frequency, and context of the question, e.g., what has been said before, what the learner is working on, or what is going on at the moment in a simulation. *Model integration* describes the ways an ITS infers information from questions to update its learner model.

Figure 4 suggests the different learner model components that an ITS can adapt based on the questions learners ask it. The questions potentially contain information about each learner’s *traits and states* or *misconceptions*. While specific misconceptions depend on the problem domain, the figure also shows some examples of domain-agnostic traits and states that researchers have differentiated with learner questions.

![Figure 4: Model integration. Questions can update global and domain-specific learner model data.](image)

Help-seeking researchers have used question features such as frequency, generality, and depth to identify several psychological traits and states in learners. Goal
orientation is one example. Learners can act from motivation to master new material, to outperform expectations, or merely to avoid underperforming (Harackiewicz, Barron, Pintrich, Elliot, & Thrash, 2002). Learners’ question-asking behavior can help reveal these goals (Ryan & Pintrich, 1997). In the same way, the questions learners ask can provide insight into their global self-esteem (Karabenick & Knapp, 1991). And since the act of asking questions requires monitoring and attempting to self-regulate knowledge, it represents a metacognitive skill (Aleven, McLaren, Roll, et al., 2006). Therefore, the tutor can make inferences about learners’ reflection and other abilities based on how effectively they use questions.

Many other traits and states may be possible to infer. For example, several ITSs have the ability to infer boredom by detecting certain behavior patterns. However, fewer ITSs attempt to detect frustration and floundering (Lane, 2006). Certainly something as simple as a text box to type invective into would give learners an easy channel to signal their frustration to the ITS.

In addition to traits and states, questions can also give valuable clues to learners’ misconceptions. Especially in complex simulations, trainees’ actions alone can sometimes give too little information to diagnose the root cause of poor performance (Kodaganallur et al., 2006). In such cases, any questions a trainee asks may provide crucial evidence to help identify a misconception.
2.2.3  Response Characteristics

An important part of using trainee questions is answering them. Intuitively, an ITS that allows questions but fails to answer in a timely fashion is unlikely to elicit effective help-seeking behavior. Directly answering questions benefits learners by letting them verify beliefs, address knowledge deficits, and establish common ground (Graesser & Person, 1994). Furthermore, any ability of the ITS to answer some questions independently represents reduced workload for human instructors.

Question answers are one class of pedagogical intervention, and this section will not review all of the timing and content recommendations that are already well studied in the context of ITS hints and help (e.g., Aleven, Stahl, Schworm, Fischer, & Wallace, 2003). However, two considerations inspired by studies of human question answering are also relevant to ITSs that answer questions: the social role the ITS plays when it answers a question and the role of the answer in an ongoing dialogue.

2.2.3.1  Social Role

Many ITSs offer hints and help to learners, so their relationship with learners is similar to that of a human teacher or mentor. Likewise, answering questions is a task that places an ITS in a specific relationship to the learner.

This section describes three social roles an ITS can play when answering a learner’s question. These roles are broad categories based on Moore’s (1989) taxonomy of interactions in effective learning. The roles, drawn as a Venn diagram in Figure 5, are authority, peer, and reference. This section describes how answering questions in
different ways might place an ITS in different social roles and thus change how learners ask questions.

One possible source of question answers is an authority, such as a teacher, trainer, or other expert. A second help source is a peer, such as another trainee participating in the same simulation. Thirdly, answers can come in the form of reference material that is not mediated by a person, such as a manual.

The role an ITS presents when responding to a question can change learners’ estimates of what to ask and how to ask their questions. For example, a learner may include more explanation when asking a peer a question than when asking an expert (Bromme, Rambow, & Nückles, 2001). When addressing an authority, learners may use more polite phrasing (Puustinen, Volckaert-Legrier, Coquin, & Bernicot, 2009).
There is precedent for ITSs to purposefully play a social role. For example, researchers study relationship manipulation explicitly when they equip an ITS with a pedagogical agent, a virtual person who helps learners in ways a teacher (e.g., Conati & Zhou, 2004) or peer learner (e.g., Burleson, 2006) might. These social agents can both improve performance and engender a positive attitude about learning (Lester et al., 1997).

ITSs very often offer learners hints and corrections. Since the ITS possesses (presumably correct) knowledge and instructs the learner, these interactions often place the ITS in the authority role. Alternatively, an ITS could also give hints to the learner as a peer by couching them in more deferential terms. Authorities and peers have beliefs about what a learner knows or can do. When they answer questions, they can empathize with frustration, share in satisfaction, or even act as role models (Burleson, 2006).

In the context of ITSs answering questions, the choice to play the role of an authority or peer is an interesting one. When interacting with humans, learners sometimes avoid asking questions to avoid displaying ignorance or, in the case of asking a peer, incurring a debt (Fisher, Nadler, & Whitcher-Alagna, 1982). This suggests that peer or authority responses from an ITS may trigger some of the same reluctance, and has led to research aimed at mitigating such negative affect (e.g., Wang & Johnson, 2008).

Even when they are playing the role of an authority or peer most ITSs do not hide the fact that they are software programs, a fact which might have the side effect of making ITSs less socially threatening than human teachers. On the other hand, a less typical mechanism for answering questions as an authority or peer might actually be relaying learners’ questions to a human. Such a method might help in situations where
the ITS fails to parse and “understand” some questions. A hypothetical ITS that used humans to answer questions would share some characteristics with collaborative learning or distance learning, but it would additionally have the potential to interpret some questions as learner model inputs.

When an ITS’s answers do not display any intentionality at all, learners may perceive it not as help but as a reference. Reference responses are characterized by their appearance of generality and lack of adaptation to the situation at hand. Even though an ITS might be personalizing reference responses internally, the learner does not perceive the answer to be mediated by an intentional agent, even an artificial one. An example is presenting a textbook page that is relevant to a learner’s question (e.g., Graesser et al., 2004). Because reference-type responses might sidestep questions of reconstructing natural language and context, they might be easier to implement. However, learners may not use reference help as much as personalized help (Aleven & Koedinger, 2000; Graesser et al., 2004), or their questions might devolve into typical web queries, i.e., lists of noun phrases (Broder, 2002).

2.2.3.2 Dialogue Context

In ITSs, question answering can take place with varying amounts of dialogue context. An ITS can answer each question separately from previous questions, can engage in an ongoing dialog, or can even create a multi-agent conversation involving more than one artificial personality or real human. The choice of dialogue context has the
potential to change the learner’s experience, for example by affecting perceived realism or feeling of presence (Chertoff, Schatz, McDaniel, & Bowers, 2008).

Dialogue management, including the task of remembering what has been said before, is a topic of ongoing research. One field using dialogue context is that of conversational agents or chatbots. Although widely used general-purpose chatbots simplify the dialogue task by processing each conversational line separately (Shah, 2006), it is possible to maintain long-term context by limiting the discussion domain (Allen et al., 1994). Interactive question answering systems also limit their domain so as to answer questions in natural language (e.g., Varges, Weng, & Pon-Barry, 2008). A second method for an ITS to answer questions with context awareness might be to constrain dialogue options to a tree, like a telephone menu system does. As for multi-agent conversations, a step in that direction is an agent that can answer questions on a multi-user forum by referring back to previous posts (D. Feng, Shaw, Kim, & Hovy, 2006).

2.2.3.3 Other Pedagogical Concerns

Finally, in addition to social role and dialogue context, answering questions also involves considerations that are familiar to most ITS practitioners from designing hints and other help vectors. For example, answers can be immediate or delayed (Butler & Winne, 1995). There are pedagogical considerations of which problems to focus on when there are more than one (J. R. Anderson, 1993). Also, answers need the right timing and detail level to avoid distraction or cognitive overload (Koedinger & Aleven, 2007). These
question response considerations have parallels in existing research on ITS intervention design. As mentioned, such issues have received substantive attention in the mainstream ITS literature and will not be further outlined in the present section. Interested readers should see Aleven et al. (2003) for more details.

2.2.4 Learner Questions in Existing ITSs

Many ITSs that have a detailed student model are able to customize the help they offer learners, such as hints about the next step in a problem. Typically, the ITS displays help either with no request, such as when the student makes a mistake, or with a constrained help request, such as clicking a hint button (e.g., Koedinger & Aleven, 2007). In some ITSs, the students’ hint usage updates the student model, for example decreasing the comprehension estimate (e.g., Ainsworth & Grimshaw, 2004) or even acting as evidence of cheating (e.g., Baker, Corbett, & Aleven, 2008). In contrast to inquiry modeling, these ITSs model learners based only on the existence or frequency of help requests, not their content. However, a few ITSs do offer freer question interfaces or draw richer information from help-seeking interactions.

STEVE (Rickel & Johnson, 1999) is a simulator-based ITS that teaches tasks such as maintaining an air compressor. A speech recognition interface lets the trainee ask three types of questions: “What is the next step,” “How do I do that,” or “Why?” The questions are answered by a Soar module that plans in real time how to complete the domain task. However, the ITS does not use the extra information from its freer question interface to
better model trainees. Further, STEVE’s effect on learning has not been reported, nor has the extent to which trainees actually ask the ITS questions.

Autotutor (Graesser et al., 2004; Graesser et al., 1999) is an ITS that teaches physics. Autotutor uses a text-based or spoken interface to carry on a dialogue with the student, making student utterances the behaviors the ITS evaluates. Since the student can freely type or speak his or her part of the dialog, students can ask the ITS questions to learn about the material. The ITS answers questions in a reference role by performing pattern-matching on students’ questions to categorize their type and content, searching a text for related sections, and presenting the top five matching sections for the student to read. The ITS does not use the students’ questions for modeling or for changing the dialogue, and Autotutor’s authors state that students do not often ask questions of the ITS. Studies have shown Autotutor has a positive effect on learning, especially on deep understanding of the topic (Graesser et al., 2004).

Quantum Simulations, Inc. is a commercial enterprise building ITSs that teach subjects such as math, chemistry, and accounting (B. G. Johnson et al., 2009). Their tutors constrain student progress through each problem by referencing a framework of steps that must be followed. As the student works on each step, an internal production rule set models the student’s thinking and selects questions the student might have, displaying them in a menu. Subject-matter experts write the questions and accompanying short answers. One such ITS has demonstrated a positive effect on learning, and anecdotal evidence suggests that students in that experiment did use the question facility (B. G. Johnson et al., 2009).
METTLE (Domeshek, 2008) is an ITS that trains medical doctors in disease diagnosis. Trainees interact with the training environment mainly by asking questions of simulated patients and doctors, so question responses are in the peer role. The trainee can freely speak or type questions to simulated agents, and since these questions are in-character behaviors they always update the learner model. In contrast, METTLE’s facility for asking questions that are instrumental to learning, rather than in-character behaviors, is limited to choosing from a menu of “What next,” “How,” and “Why?” These questions are answered outside the simulator context and do not update the ITS’s trainee model.

2.2.5 Inquiry Modeling Summary

Allowing learners to ask questions of an ITS has potential advantages for learners, instructors, and ITS efficacy. Seeking help plays an important role in instruction, letting learners fill in their knowledge gaps and exercise metacognition. Answering frequently asked questions without taking up a human’s time can lessen instructors’ workload and give them better feedback about their students. Finally, questions in human tutoring contexts have been used to diagnose learners’ knowledge, affect, or other cognitive states. If an ITS can answer questions it would be able to use this information to improve its user model.

The present section drew together interdisciplinary research to build background information about questions learners ask during instruction, collect evidence showing the benefits of help seeking, and suggest different ways an ITS can input questions, answer
them, and then use them to update its student model. Based on this previous research, the present dissertation explores inquiry modeling in a real-world ITS.

2.3 POMDPs for Modeling and Planning

Partially observable Markov decision processes (POMDPs) and their potential for driving intelligent tutoring systems were introduced in Section 1.3. Although POMDPs have not been widely used as ITS user models to date, several ITSs employ Bayesian user models. The first part of this section discusses successes of different Bayesian models in intelligent tutors. Various specialized POMDP representations have also been used in modeling large problems other than tutoring or training. The second part of this section discusses published approaches to make large problems tractable. Although different algorithms for learning policies are not a focus of this dissertation, the third part of this section discusses some approaches, including the approach used in the present research.

2.3.1 Bayesian ITS User Models

POMDP ITSs build on the successes of other Bayesian models in intelligent tutors. Bayesian models can handle uncertainty and recover from errors, infer hidden state values from evidence, and allow machine learning of some model components during development (Pearl, 1988). The usefulness of increasingly complex Bayesian models in ITSs suggests POMDPs could also control ITSs successfully.

As an initial example OLAE, an assessment tool and precursor to the Andes physics tutor, observes individual steps of learners’ work to infer their domain knowledge mastery (VanLehn & Martin, 1998). Similarly, the ITS Ecolab observes learners’ help
use to predict both topic mastery and readiness to learn each new topic (Luckin & du Boulay, 1999). In both models, subject-matter experts designed networks by hand that used Bayesian methods to interpret assessments.

The Bayesian model architecture also lets developers apply machine learning to speed tutor development and refine accurate models from real learner data. For example, the high-school math tutor Wayang Outpost (Arroyo, Beal, Murray, Walles, & Woolf, 2004) observes help use to form mastery estimates. In this system, machine learning from experimental data helped define model parameters (Ferguson, Arroyo, Mahadevan, Woolf, & Barto, 2006).

In the elementary grammar tutor CAPIT (Mayo & Mitrovic, 2001), Bayesian methods inform a still larger portion of the tutor behaviors. Whereas previous Bayesian-model tutors selected pedagogical actions with heuristics or hand-written rules, CAPIT can predict action outcomes with its learner model. CAPIT also modifies its Bayesian model updates with a simple discount function to reflect the fact that learners’ knowledge changes over time.

Dynamic Bayesian networks (DBNs) are Bayesian networks that can model changes over time. An early version of the Andes physics tutor (Conati, Gertner, & VanLehn, 2002) used a DBN, and the Prime Climb math tutor (Conati, 2002; Conati & Maclaren, 2009) models learner goals and affective states with a similar, DBN-like construct that substitutes a non-Bayesian update method for better scaling.

Finally, a coin tutor for elementary students (Theocharous, Butko, & Philipose, 2010) uses a full-fledged POMDP learner model. For this prototype, problem state is
assumed to be completely observable, concepts are ordered strictly linearly, and only one cognitive state is modeled (attention / distraction). Another instance of the tutor (Theocharous, Beckwith, Butko, & Philipose, 2009) has a six-state learner model with factorization (Boutilier, Dean, & Hanks, 1999) to address scaling problems, creating several small POMDPs that can be selected by a second, hierarchical POMDP. A POMDP that controls lesson selection in a military training setting has 11 states (Levchuk, Shebilske, & Freeman, in press; Shebilske, Gildea, Freeman, & Levchuk, 2009). Compared to previous Bayesian ITSs, these POMDP ITSs model fewer facts about each learner.

In summary, Bayesian models allow diagnosis despite uncertainty and the possibility to base parameters and structure on expert requirements, empirical data, or both. Dynamic solutions such as DBNs and POMDPs can additionally model change over time. While DBNs estimate hidden states and require separate, hand-written rules to act on the diagnoses, optimal intervention planning is integral in POMDPs. However, existing POMDP ITSs only model relatively few features or learner states.

Recurring problems with Bayesian ITS user models in general have been the difficulty of learning or specifying large numbers of parameters to build a model (Ott et al., 2004), and the difficulty of gathering sufficient evidence from performance during tutoring (VanLehn & Niu, 2001). The new problem representations discussed in the present dissertation use tutoring-specific assumptions to help decrease the number of model parameters that must be specified. Inquiry modeling, in turn, can help add new model inputs during tutoring by drawing information from questions learners ask.
2.3.2 Tractable POMDPs

POMDPs are known to scale poorly with lossless or enumerative problem representations. Representations are *lossless* when they explicitly encode all possible information of some type, rather than using domain-specific knowledge to avoid storing unimportant information. Representations are *enumerative* when they encode all information individually, instead of leveraging known problem structure regularities to save on storage or computation. In contrast to these naïve representations, a *lossy* representation might remove from the model some states that do not impact performance in the problem domain. A *compositional* representation might reduce the number of states by introducing a single structure to represent several different states that share some feature and can be treated the same in the problem domain.

The number of states naïve state representations must contain grows exponentially with the number of independent variables to describe, quickly growing impractical. For example, a moderately sized problem might require ten binary values to describe a system. An enumerative and lossless state representation would then require $|S| = 2^{10} = 1,024$ separate states to store all possible combinations of system values. Likewise, state transition probabilities, observation emission probabilities, and rewards all depend on system state. Therefore naïve representations of these values in a POMDP would have exponentially increasing numbers of parameters to store (and either specify or learn) in at least three more places, up to $|T| = |S|^2 \times |A|$, $|O| = |S| \times |A| \times |\Omega|$, and $|R| = |S|^2 \times |A| \times |\Omega|$. During the naïve POMDP’s policy learning and application, beliefs and policies would have similar space requirements.
To address tractability concerns, POMDP states and transitions can themselves be represented by a DBN (Boutilier & Poole, 1996). DBN nodes represent states and arcs represent state transitions. The DBN only has to represent the random variables that underlie the POMDP states—in the above example, the ten binary values that combine to describe the problem fully, rather than the 1,024 states required for an enumerated representation. The DBN is not solved but only represents symmetries in the state and transition tables. If enough states are independent of each other a DBN can represent the underlying features quite compactly. The process of identifying and representing these underlying independent features is called state-space factorization.

Other compact lossless state representations are also possible. Hierarchical structure in a particular problem can be exploited to build hierarchical POMDPs (Theocharous & Mahadevan, 2002). Groups of states that function together independently are each represented as separate POMDPs, with higher-level POMDPs to select which lower-level one to run. Separating the POMDPs decreases the number of dependencies to consider. In other problems, it may be possible to identify certain states that can be observed reliably enough that the complexity of a POMDP is not necessary to represent them. Again, properties of the problem being modeled allow a state representation that does not contain some dependencies. Modified POMDPs that represent a subset of their states as fully observable are called mixed-observability MDPs, or MOMDPs (Ong, Png, Hsu, & Lee, 2009).

State aggregation decreases the number of states in a POMDP by combining states that have the same present and future reward values, even when they have different
underlying meanings. Lossy aggregations combine states with approximately similar values. Aggregations can be chosen in advance to align with actions and rewards (Boutilier & Poole, 1996), iteratively during policy search (Z. Feng & Hansen, 2004), or using reinforcement learning to decide which states should be represented over the course of many runs (McCallum, 2002).

State transitions, observation emissions, and reward functions can also take advantage of problem properties for more compact storage or easier calculation. Reward functions can be represented as sums of simpler functions (Bacchus & Grove, 1995). State transitions and observation emissions can be represented functionally, such as with decision trees (Boutilier & Poole, 1996). The matrices can also be learned online, as the controller uses the POMDP (Atrash & Pineau, 2010). Approximate representations are also possible to build, such as Poupart’s value-directed linear compression (2005), which can use iterative error minimization to find a system of representation functions.

Finally, the sets of actions and observations themselves might be represented compactly. Making these sets smaller potentially improves policy search by reducing the branching factor at each possible step in a policy. The actions represented in a POMDP can be made less numerous by grouping several atomic actions together into macro-actions that dictate several turns in a row (Hauskrecht, Meuleau, Boutilier, Kaelbling, & Dean, 1998). Appropriate macro-actions to suit a particular problem can be specified or built with machine learning (He, Brunskill, & Roy, 2010). POMDP observations, like states, can be factorized into independent underlying components or aggregated into
groups of observations that all have the same effect on planning, for example because they give the same information about the current state (Hoey & Poupart, 2005).

2.3.3 Tractable Policy Search

POMDP policies are functions that choose an action to maximize reward given a particular belief. Since POMDP beliefs represent the probability the system is in each possible state, belief space has $|S| - 1$ continuous dimensions. The problem of efficiently searching for policies despite this high dimensionality makes state-space size explosion an even bigger problem. Furthermore, the number of possible policies is exponentially related to the number of observations, possible actions, and planning horizon. This section describes several previously published approaches to policy search, including the approach used in this dissertation, SARSOP (Kurniawati, Hsu, & Lee, 2008). There are three main methods for solving POMDPs: conversion to another representation, policy iteration, or value iteration.

DBNs can be made equivalent to entire POMDPs, representing not just their states and transitions but all information about previous states that would otherwise be encoded in a POMDP’s belief (Murphy, 2002; Theocharous, Murphy, & Kaelbling, 2004). Such DBNs can easily become quite large, requiring as many time-slices as the POMDP history is long. Learning model parameters in the resulting DBNs is then equivalent to finding a policy.

Policy iteration refers to explicitly representing policies with some decision-making construct and searching for the best one; convergence is reached when the best
policy no longer changes between iterations. For example, if finite state machines of a set size represent policies, then they can improve with gradient descent (Baxter & Bartlett, 2001) or reinforcement learning (Wierstra, Foerster, Peters, & Schmidhuber, 2007). POMDP policies can also be represented by neural networks and learned through artificial evolution (Gomez & Schmidhuber, 2005). These three search algorithms suffer from local optima, meaning that they are not guaranteed to find the best policy. Branch and bound is another algorithm for policy search (Meuleau, Kim, Kaelbling, & Cassandra, 1999). Applied to finite-state-machine policies, it moves the scaling problem from belief-space size to policy-space size, meaning the algorithm works best in searching for simple policies. A hybrid method designed by Poupart (2005), bounded policy iteration, combines branch and bound with local gradient descent.

Value iteration refers to estimating the values of each possible action given a particular belief. The resulting policy is then to select the action with the highest projected value. To address continuous belief space, policies can be represented compactly with alpha vectors, linear functions representing action values that together form a piecewise-linear convex surface in belief space representing the value of an optimal choice (Smallwood & Sondik, 1973). Without any optimization, this requires searching an exponential number of points in belief space (Pineau, Gordon, & Thrun, 2003). Pruning dominated points (Cassandra, Littman, & Zhang, 1997) reduces the branching factor and thus the exponential base, although algorithm complexity remains exponential.
Value iteration can be made practical by sampling only a small number of points in belief space. Methods for selecting the locations to sample include examining a regular grid of points (Lovejoy, 1991), randomly choosing from among reachable beliefs only (Pineau et al., 2003), and estimating points that lie in the path of approximately optimal behavior (Kurniawati et al., 2008). An algorithm Kurniawati and colleagues (2008) described that alternates sampling belief space and approximating optimal behavior is Successive Approximations of the Reachable Space under Optimal Policies (SARSOP). This algorithm represents an automation of applying domain knowledge that is already encoded in the POMDP. SARSOP is the policy search algorithm used in the present dissertation work, because it is able to accommodate large state spaces such as the ones an ITS models.

2.3.4 POMDP Review Summary

This section reviewed advances that have been made to make POMDP problem representations and policy search more tractable, and applications of Bayesian user models in existing ITSs. Although much effort has been dedicated to improving general methods for alleviating the exponential costs associated with both representation and solution in POMDPs, general methods must be able to handle possibilities that may never apply in a particular problem domain. There remains a need—practically a requirement—for domain-specific enhancements to make POMDPs useful in solving real-world problems. The impracticality of unenhanced POMDPs for large problems is likely to help explain why POMDPs have not been used to drive full-scale ITSs. The present
dissertation studies two specialized representations that let POMDPs model problems of the scale that ITSs must face.
CHAPTER 3: MAKING POMDPS PRACTICAL IN AN ITS

This chapter describes two compression schemes that make POMDPs capable of representing problems as large as those real-world ITSs confront. The first, *state queues*, compress a POMDP’s representation of system states, while the second, *observation chains*, compress a POMDP’s observation representation. Both compression schemes make use of properties of the ITS problem to discard some data for greater scalability.

State queues and observation chains are designed to compress a wide variety of possible problem representations. The following two subsections introduce a simple example of a user model, $M$, with a specific state, action, and observation structure. The components in the example model can, in turn, be assigned many interpretations. The present section refers to a specific hypothetical ITS, a basic arithmetic tutor, to explain the structure and representation of $M$ more clearly. However, similar structures could also represent the states, actions, and observations that make up any number of ITSs. For example, the structure of $M$ is the same as the simulator training learner model evaluated in this dissertation’s formative and summative studies.

3.1 POMDP State-Space Compression

3.1.1 An Example User Model

This section explains how POMDP hidden states can model the current mental status of a particular user. A user’s mental status is made up of *knowledge states* and *cognitive states*.
Knowledge states describe learner knowledge. $K$ is the domain-specific set of specific misconceptions or facts or competencies the user has not grasped, together termed *gaps*. The intelligent tutor should act to discover and remove each gap. For example, in an arithmetic tutor one missing skill in $K$ might be “the user does not recall with sufficient automaticity that $2 + 2 = 4$,” an example of missing knowledge in $K$ might be “the user does not know the meaning of *commutative*,” and a misconception in $K$ might be “the user adds columns without carrying any digits.”

The set $C$ of cognitive states represent transient or permanent properties of a specific user that change action efficacy. Cognitive states can also explain observations.

Cognitive states are not domain-specific, although the domain might determine which states are important to track. In the example user model $M$, the set $C$ includes boredom, confusion, frustration, and flow. Previous research supplies estimates for how these affective states might change learning effectiveness (Craig et al., 2004). Other ITSs may also model other states such as goal orientation, workload, interruptibility, personality, or demographic groups. For example, an arithmetic tutor might need to present word problems with different settings depending on whether users are children or adult learners.

Many hints or other actions could tutor any particular gap. In $M$, the ITS’s POMDP decides which gap to address (or none). A pedagogical module, separate from the POMDP, is posited to choose an intervention that tutors the target gap effectively in the current context. So, an arithmetic tutor with a structure like $M$ would have access to actions approximately corresponding to each member in its $K$, such as “present practice
for quicker access to the math fact $2 + 2$” or “remind the user about the definition of *commutative.*” The pedagogical module would then select an appropriate intervention to carry out the chosen action.

As an alternative to the external pedagogical module in $M$’s example, a POMDP could also assume fine control over hints and actions, possibly personalizing them according to its knowledge of a user’s cognitive and knowledge states. However, the efficacy for each possible intervention would then need to be estimated.

Whether the POMDP controls interventions directly or through a pedagogical module, each POMDP action $i$ has a chance to correct each gap $j$ with base probability $a_{ij}$. For example, the action “remind the user about the meaning of *commutative*” might have a 50% probability of clearing the gap “user does not know the meaning of *commutative*,” and a 0% probability of clearing any other gaps. On the other hand, it might just as well be appropriate to assign the same action a 10% chance of clearing the gap “user does not know the meaning of *associative*,” on the grounds that the user might have simply confused the two words. The interventions available to the ITS determine the values of $a$, and they must be either learned or set by a subject-matter expert.

For simplicity, no actions in $M$ introduce new gaps, a decision which also means that in $M$ information once learned will not be forgotten. In this dissertation’s instructional domain, training takes place over a short time period so forgetting learned information is less likely. However, in many domains it is not realistic to assume actions cannot create gaps, so future work should relax this assumption by adding transitions in
that model misunderstanding or forgetting. The extra transitions will increase the number of parameters to define but not change $M$ qualitatively.

The user’s cognitive state may further increase or decrease the probability of correcting gaps according to a modifying function $f_c: [0,1] \rightarrow [0,1]$. In $M$, values for $f_c$ approximate trends empirically observed during tutoring (Craig et al., 2004). Transitions between the cognitive states are also based on empirical data (Baker, Rodrigo, & Xolocotzin, 2007; D’Mello, Taylor, & Graesser, 2007). Actions that do not succeed in improving the knowledge state have higher probability to trigger a negative affective state (Robison, McQuiggan, & Lester, 2009). In the arithmetic tutor example, a bored user is more likely to click past a hint message without reading it. And presenting a definition of *commutative* when the user is really wondering what *associative* means, or even an obscure and unhelpful definition of *associative*, is more likely to leave the user in a frustrated state. Because of the cognitive state modifications in $f_c$, an ITS might improve its overall performance by not intervening when no intervention is likely to help, such as when there is too much uncertainty about the user’s state.

Action types not included in the example model $M$, such as knowledge probes (for example, presenting a math quiz) or actions targeting cognitive states alone (for example, displaying an encouraging message), are also possible to represent with a POMDP.

### 3.1.2 Tutoring Problem Characteristics

Often, tutoring problems have three characteristics that can be exploited to compress their state-space representations.
First, in many cases certain knowledge gaps or misconceptions are difficult to address in the presence of other gaps. For example, when a person learns algebra it is difficult to understand exponentiation before multiplication. Multiplication in turn is difficult to comprehend without a grasp of addition. This relationship property allows instructors in general to assign a partial ordering over the misconceptions or gaps they wish to address. They make sure learners grasp the fundamentals before introducing topics that progressively build on the learners’ knowledge.

To reflect the way the presence of one gap, \( k \), can change the possibility of removing a gap \( j \), a final term \( d_{jk} : [0,1] \) is added to \( M \). The probability that an action \( i \) will clear gap \( j \) when any other gaps \( k \) are present, then, becomes \( f_i(a_{ij} \prod_{k \neq j} (1 - d_{jk})) \). Values of \( d_{jk} \) near 1 indicate that for pedagogical reasons gap \( j \) “depends on” gap \( k \), and it is difficult to remove gap \( j \) as long as gap \( k \) exists.

Second, in tutoring problems each action often affects a small proportion of \( K \). Presenting a lesson about addition will not give a student a sudden understanding of multiplication. Furthermore, in ITS model design, interventions often target individual gaps. For example, an ITS that models a specific misconception about carrying during addition is likely to contain an intervention targeting that misconception. In contrast, an ITS that models several addition misconceptions but addresses all of them by re-teaching the entire addition lesson does not meet this criterion. The second characteristic holds true for ITS problems where \( a_{ij} = 0 \) for relatively many combinations of \( i \) and \( j \).

Third, the presence or absence of knowledge gaps in the initial knowledge state can be close to independent, that is, with approximately uniform co-occurrence
probabilities. Finding that a learner has a misconception about carrying in addition does not make it more or less likely the learner will need drilling in multiplication facts. This characteristic is the most restrictive, in that it is probably less common in the set of all tutoring problems than the other two, but such problems do exist. For example, in cases when a tutor is teaching new material, all knowledge gaps will be likely to exist initially, satisfying the uniform co-occurrence property.

3.1.3 The State Queue Representation

In tutoring problems, a partial ordering over K exists and can be discovered, for example by interviewing subject-matter experts. By reordering the members of K to minimize the values in d where j < k and breaking ties arbitrarily, it is further possible to choose a strict total ordering over the knowledge states. This ordering does not necessarily correspond to the POMDP’s optimal action sequence, because of other considerations such as current cognitive states, initial gap likelihoods, or action efficacies. However, it gives a heuristic for reducing the number of states a POMDP must track at once.

A state queue is an alternative knowledge state representation. Rather than maintain beliefs about the presence or absence of all knowledge gaps at once, a state queue only maintains a belief about the presence or absence of one gap, the one with the highest priority. Gaps with lower priority are assumed to be present with the initial probability until the queue reaches their turn, and POMDP observations of their presence or absence are ignored except insofar as they provide evidence about the priority gap.
POMDP actions attempt to move down the queue to lower-priority states until reaching a terminal state with no gaps.

The state space in a POMDP using a state queue is $S = C \times (K \cup \{done\})$, with $done$ a state representing the absence of all gaps. Whereas an enumerated state representation grows exponentially with the number of knowledge states to tutor, a state queue grows only linearly.

State queuing places tight constraints on POMDPs. However, these constraints may lead to approximately equivalent policies and outcomes on ITS problems, with which they align. If ITSs can only tutor one gap at a time and actions only change the belief in one dimension at a time, it may be possible to ignore information about dimensions besides the one to move in first. Finally, the highly informative observations that are possible in the ITS domain may ameliorate the possibility of discarding some.

### 3.2 POMDP Observation Compression

#### 3.2.1 Example User Model Continuation

Like actions, the proposed representations are also agnostic to ITS observation content. In the example user model $M$, an external assessment module is posited to preprocess observations for transmission to the POMDP, a design decision motivated by the typically complex and non-probabilistic rules that map user behaviors into performance assessments aligned with the state space.

The assessment module preprocesses ITS observations of user behavior into POMDP observations of assessment lists. POMDP observations are indicators that
certain gaps are present or absent based on tutee behavior at that moment. The POMDP’s task is to find patterns in those assessments and transform them into a coherent diagnosis in its belief state.

In an arithmetic tutoring situation, a tutor might ask a user the question “2 x (3 + 4) = ?” and receive the incorrect response “10.” A likely reason for this particular wrong answer is that the user actually solved “(2 x 3) + 4,” but in general it is infeasible to program a POMDP with every specific question and possible user answer at such a level of granularity. Instead, the assessment module would map observations of specific user behaviors such as this incorrect answer into one or more assessments that the POMDP observes. The mapping might be accomplished through model tracing (J. R. Anderson, 1993) or another accepted means. The POMDP observation, then, would convey a single assessment module output such as “the user did not distribute multiplication within parentheses properly.” The POMDP could use such an observation to increase the probability of some diagnoses, for example “the user does not know where parentheses stand in the order of operations or the user does not know how to distribute multiplication,” while possibly decreasing the probability of other beliefs.

In $M$, each POMDP observation contains assessment information indicating the presence of one gap and the absence of one gap in the current knowledge state. A non-informative blank assessment can also appear in the present or the absent dimension, meaning that an observation might not include any information about that dimension. The ITS can incorrectly assess absent gaps as present when a tutee slips, and present gaps as absent when the tutee makes a guess (Norman, 1981). Although real-world observations
would probably also include information about cognitive states, observations in this simplified model only pertain to knowledge states.

3.2.2 Problem Characteristics and the Observation Chain Representation

A key property of ITS problems is that POMDP observations can be constructed to convey information in approximately orthogonal dimensions. For example, a learner’s success on a math problem that requires addition, multiplication, and exponentiation could form evidence for mastery of all the competencies needed to complete the task. Conversely, specific kinds of mistakes could be evidence the student has mastered some math skills but certain misconceptions still exist. The various skill assessments represent independent dimensions that are all transmitted to the ITS by observing a single event.

An example observation construction that describes multiple dimensions is the definition of an observation in $M$, which contains information about one gap that is present and one gap that is absent. These two components are orthogonal, and in fact it is possible to observe the presence and the absence of the same gap simultaneously, though one would be untrue. Similarly, evidence of multiple gaps could be gleaned from one observation. For instance, the arithmetic tutor assessment that “the user did not properly distribute multiplication within parentheses” could be interpreted as bearing information about two separate features, the user’s knowledge of order of operations and ability to distribute. Many other ITS problems may also communicate evidence about several features in one observation, such as a user’s performance on a complex simulator task that requires many competencies to complete.
An opportunity to compress observations lies in the fact that ITS observations can contain information about multiple orthogonal dimensions. Such observations can be serialized into multiple observations that each contain non-orthogonal information. A process for accomplishing this is *observation chaining*, which compresses observations when emission probabilities of different dimensions are independent, conditioned on the underlying state.

Under observation chaining, observations decompose into families of components. A family contains a fixed set of components that together are equivalent to the original observation. Decompositions are chosen so that emission probabilities within components are preserved, but emission probabilities between components are independent.

As an example, imagine a subset of $\Omega$ that informs a POMDP about the values of two independent binary variables, $X$ and $Y$. In the arithmetic tutor, $X$ might indicate “used correct order of operations” and $Y$ “distributed correctly.” To report these values in one observation, $\Omega$ might contain four elements $\{XY, X'Y, XY', X'Y'\}$. An observation chain would replace these elements with one family of two components, $\{X, X', Y, Y'\}$. Then when one of the original elements would be observed, a chain of two equivalent components is observed instead. Whenever one observation completely describes more than two independent features, a chain representation will add fewer elements to $\Omega$ than an enumerated one.

Chains can be of arbitrary length, so a token *end* is added in $\Omega$ to signal the end of a chain. In $M$, the simplified example user model, assessments a POMDP observes can
signal the presence of a particular knowledge gap, its absence, or neither. A blank chain element that conveys no information signals an observation that is uninformative about any gap. With observation chaining, \( \Omega = (K \cup (\text{Soh, Blank, \\& Miller})) \cup (K' \cup (\text{Soh et al.}')) \cup \{\text{end}\} \).

In POMDPs with observation chaining, \( S \) contains an unlocked and a locked version of each state. Observing the end token moves the system into an unlocked state, while any other observation sets the equivalent locked state. Transitions from unlocked states are the same as in an equivalent POMDP without observation chains, while locked states always transition back to themselves. Therefore, the ITS can observe any number of chain elements before it recommends another action, and the POMDP controller does not need to be rewritten to use the different chain lengths. Observation chains make \( \Omega \) scale better with the number of orthogonal observation features, at the cost of doubling \(|S|\).

Observation chains are similar to the plan Hoey and Poupart (2005) suggest for reducing conditionally independent observation dimensionality. However, where that representation adds a counter to the state space and transmits observations in order, the lock-unlock scheme does not impose an order on chain elements and requires no counter. Therefore, reducing observation space by some factor does not require increasing state space by the same factor, as under Hoey’s and Poupart’s algorithm. Instead, observation chains deliver exponential observation-space improvements with only a one-time doubling of state space cardinality.
3.3 POMDP ITS Representation Summary

This chapter introduced an example user model structure that lets a POMDP represent an ITS problem, and it discussed two compression schemes that make large problems possible to represent with such structures.

The representations discussed in this paper have advantages in some circumstances over other POMDP simplifications discussed in Section 2.3. Compared to existing examples of state-factored or hierarchical POMDP ITSs (Theocharous et al., 2009; Theocharous et al., 2010), state queues better preserve relationships between cognitive states or other modes that form the dividing lines in factored POMDPs, at the cost of some knowledge state information. State queues can be used in conjunction with other state factoring schemes. Compared to macro-actions that shorten lookahead horizons by grouping actions (He et al., 2010), the compressed representations do not limit observations or constrain the range of tutoring options. While macro-actions may be unnecessary with the present compressed representations which also make policy search easier, they can be used together. Finally, an alternative observation compression scheme is the one proposed by Hoey and Poupart (2005), in which all observations that lead to the same policy are aggregated. This scheme can integrate with the smaller state spaces of state queues, in place of observation chains. However, that algorithm requires data to learn the aggregation and therefore works best with a small number of possible plans, whereas ITSs usually have many actions which can be combined into even more possible plans, making compression schemes that do not take advantage of domain knowledge difficult to implement.
There is a limitation to the compression strategies: some data are discarded. The information discarded by state queues and observation chains, however, is theorized to have little impact on outcomes for certain classes of real-world problems, including the problem of adaptive tutoring. In fact, experiments in the present dissertation demonstrate that the new representations let a POMDP effectively control an ITS.
CHAPTER 4: THE CFF INSTRUCTIONAL DOMAIN

An advantage of the $M$ model structure described in the previous chapter is that it can be generalized to intelligent tutoring in many domains. For this dissertation, a model was developed for a simplified military task. The present chapter describes this use case instructional domain.

The task to be trained, known as call for fire (CFF), is performed in the real world by United States Army and Marine Corps personnel called Forward Observers (FOs). These personnel work on the front lines of battle to observe enemy positions and direct attacks from allied units. After an FO transmits the locations and descriptions of enemies, distant artillery and other units can target the enemies with precise fire that is more likely to be effective and less likely to harm friendly units or civilians in the area. Figure 6 presents a schematic representation of a CFF scenario.

Figure 6: A schematic representation of a Call for Fire scenario. Enemy units, left, attack friendly units and an encampment. Friendly artillery, off-screen, will fire on the enemies after the Forward Observer makes a call for fire.
Actions in the CFF domain include dialogues between the FO and the remote artillery. Each CFF dialogue must contain accurate information about the FO’s location, the target’s relative position, the type of ammunition to use, and whether to fire one shell or several. If some part of this information is incorrect, the artillery’s fire will be ineffective. An FO must also decide which units to engage and in what order. Errors in any part of the CFF dialogue can have different underlying causes, such as incorrectly identifying a unit’s type or misremembering the prescribed method to attack that unit.

4.1 FOPCSIM

In the U.S. Marine Corps, FOs may train to perform the CFF task while in garrison or deployed with the Deployable Virtual Training Environment (DVTE), a laptop-based suite of training programs (Bailey & Armstrong, 2002). One program that trains the CFF task is the Forward Observer Personal Computer SIMulator (FOPCSIM). FOPCSIM is a first-person simulation, meaning that the laptop screen presents the same view a person would have while performing the CFF task in the field. Trainees move, look around, and interact with simulated tools by manipulating a mouse and keyboard. Unfortunately, as software in active military use, DVTE and FOPCSIM are listed on the United States Munitions List (USML), so no screenshots may be provided here. Instead, Figure 7 presents a schematic representation of a typical FOPCSIM screen.
Figure 7: A schematic representation of the FOPCSIM interface. A realistic image fills the screen, and overlaying it are icons to activate simulated tools. Activating the radio tool causes several text boxes and menus to overlay and obscure the screen. Trainees interact with the simulator using a keyboard and mouse (the arrow pointer at right).

When FOPCSIM starts, the trainee is presented with a first-person view of a 3-D rendered desert landscape. Overlaying the top left corner of the screen are icons representing tools the trainee can use to complete the CFF task. The trainee can swivel the viewpoint by clicking and dragging a mouse. The trainee begins the practice looking toward some targets, and all the targets are grouped near each other, although the trainee must move the viewpoint a few degrees left or right to see all the targets. The targets are displayed as Soviet military vehicles dating from the Soviet-Afghan War and are not difficult to acquire visually. Trainees do not need to move their character in the simulator other than swiveling the viewpoint, although they can do so.
Icons overlay the top left corner of the simulator screen and give access to several tools. Clicking an icon makes the corresponding tool overlay the simulator screen, and clicking again hides the tool. In order to complete CFF tasks, trainees need to use in-simulator tools representing a Precision Lightweight GPS Receiver, a Vector 21-B Common Laser Rangefinder, and a radio menu. Icons for other tools that the trainees in the present study do not use are also available including a map, a compass, and a clipboard for making notes.

For all CFF training and test scenarios, trainees must first select a target and then engage it. Target selection requires visually acquiring and identifying enemy targets, and then prioritizing them according to the threat they pose in order to select the most threatening target. Engaging the selected target requires communicating the target’s location and type to a remote fire direction center (FDC), as well as choosing the firing pattern and munition type that will most effectively engage the target. Such a transmission is itself termed a call for fire. Trainees engage one target at a time and then move on to the next target.

A series of menus replaces spoken radio communications from the trainee to the FDC. In all scenarios for this study, the role of the FDC is played by the computer. Trainees hear replies from the FDC as well as ambient sounds through a pair of headphones. To complete one call for fire, trainees must follow a complex procedure and set correct values in the radio menu for four text input fields, six dropdown menus, and seven buttons in sequence. There are also 28 other controls on the menu that do not need to be used for this particular CFF task.
4.2 Use Case Tasks

To begin a successful call for fire in the use case scenarios, trainees must first visually acquire a target. This task draws on a trainee’s existing perceptual abilities rather than an application of knowledge or skills built during practice. However, targets are not hidden and seeing them is assumed to be easy to accomplish.

After acquiring targets, trainees must distinguish which are friendly and which are enemies. Trainees are instructed that two particular vehicle types are friendly and two types are enemy targets. Trainees must remember which target types are enemies and recognize them based on visual cues, with the help of simulated visible-light optics if needed. Trainees must not attack friendly targets.

After identification, trainees must prioritize enemy targets according to the threat they pose. Half of the targets are moving in repetitive patterns and half are stationary. Moving targets are a higher priority for attack than non-moving targets. After movement, trainees should prioritize targets based on proximity to a protected area such as their own location or a friendly unit.

Target acquisition, recognition, and prioritization are grouped as target selection skills. When a trainee has selected the highest-priority enemy target, the trainee must next engage the target. Correct target engagement requires trainees to determine a target’s location, description, and the correct firing pattern and ammunition required to attack the target.

Target location is reported in terms of direction and distance from the trainee to a target. To find this information, trainees must correctly operate a rangefinder in the
simulation. They must also report their readings in the correct format through the radio menu.

In addition to target location, trainees must report descriptive information about the target to help the artillery engage it correctly. Trainees must name the type of enemy vehicle they are targeting and choose a firing pattern and munition to engage it. To correctly apply the simplified engagement rules used in this study, trainees must choose a wide spread of fire for moving targets or a single shot for stationary targets, as well as a stronger munition for heavily armored targets and a cheaper munition for lightly armored targets. Trainees choose the target description, firing pattern, and munition from dropdown menus containing options for the correct choices as well as multiple alternatives.

4.3 Call for Fire Summary

This chapter described the Call for Fire (CFF) task, a simulation environment used for training that task, and some features of the use-case scenarios employed in this study. Call for fire practice is a realistic training domain and requires a learner model to contain some complexity. The scenarios in this dissertation present simplified CFF circumstances. Real military trainees would be challenged with even more complications such as coordinating with multiple friendly agencies, using more munitions and firing patterns, and operating under adverse weather, night, or enemy fire. Such additions were not possible in the present study, but the practice environment and domain tasks
described still challenge non-specialist study participants. The need for support during the simplified practice is very real.
CHAPTER 5: FORMATIVE STUDIES

This chapter presents the findings of formative studies performed while designing an ITS for training in the CFF domain. These studies support the four research hypotheses in Section 1.4. Section 5.1 describes an anonymous questionnaire of ITS researchers and practitioners about ITS user models. This study suggested a need for a new type of learner model that is easy to develop but still produces high effects on learning, such as a planning POMDP model. Section 5.2 details a series of simulation experiments that empirically explored the experimental compression schemes’ practical effects on performance. The study showed the new representations scale well and are useful under a wide range of circumstances. Finally, Section 5.3 describes a study with human subjects that gave new information about questions learners ask during training. This study suggested ways that an inquiry modeling ITS could mine useful information from users’ questions.

5.1 Expanded and Updated ITS Development Cost Information

As described in Section 2.1, the ITS research community has produced some limited reports on system development costs, including a comparison of the same team developing two equivalent model types and a comparison of experts in their respective architectures developing equivalent models. However, publication of development time estimates remains sparse, with only a few estimates published for some model types and none at all for other widely used architectures. Further, some older published accounts may now be outdated and fail to account for advances in authoring technology.
Section 5.1 describes a questionnaire of ITS experts ($n = 11$) that helped to address these gaps in the published knowledge. The size of the survey, while too small to support detailed conclusions, nevertheless approximately doubled the number of published reports about ITS development costs. The resulting data aligned with and amplified previously published accounts, as well as contributing new cost information about model types that had not previously appeared in the literature.

5.1.1 Method

5.1.1.1 Questionnaire

An anonymous questionnaire was emailed to ITS community members in September of 2009. The text of the questionnaire appears in Appendix A.

Questionnaire respondents described their experiences on the last ITS each person worked on that was ready or almost ready to interact with learners. Therefore, participants’ memories were more recent and the data presented better reflected current modeling and authoring technology. Participants were asked to estimate the development effort in person-hours for the ITS as a whole and also for the learner model or models specifically. To calibrate the complexity of the ITS being described, participants were also asked the amount of time one learner would be expected to engage with the ITS. All questions were optional.

Participants were asked 29 additional questions, numbered 9 through 37 in Appendix A, relating to previous experiences with building specific model types.
Because of low response rates and space limitations, those questions are not discussed here.

5.1.1.2 Participants

The questionnaire was emailed to all 63 attendees of the 2009 Army Research Institute Workshop on Adaptive Training Technologies and to an additional 88 authors of publications cited in a survey of the ITS field (Folsom-Kovarik & Schatz, 2011) who did not attend the workshop. Eleven participants responded anonymously. The responses gave a varied anecdotal view of the development costs for different student models in the current state of the field.

Participants in the study came from diverse backgrounds. Of the eleven participants, five people were academics, three worked in industry, and two worked in government or military positions. Three people had worked on one or two ITSs, three had worked on three to five ITSs, and four had worked on six ITSs or more. Three people had worked on ITSs for three to six years and seven had worked on ITSs for seven years or more. One participant did not share any demographic data.

5.1.2 Results

5.1.2.1 Model architectures in current ITS development

Out of eleven participants, nine reported that the ITS he or she worked on most recently used a single learner model. Two reported using two learner models, and none reported using more than two constructs.
The models participants used included representatives from five of the six architecture categories described in the Section 2.1.2 survey. Example tracing was not represented. Note that the mention of a model type in this section does indicate current ITS research or development is using that architecture, but failure to mention a type does not indicate whether that architecture is in common use or not.

Three participants reported using model authoring tools to speed development—ASPIRE (Mitrovic et al., 2006), FlexiTrainer (Ramachandran, Remolina, & Fu, 2004), and one unnamed tool. Five participants stated they developed their entire ITS with no authoring tools. Authoring tool use did not account for a significant cost difference (two-tailed T-test).

5.1.2.2 Development cost ratios

This section relates individual experiences with building different model types. As elsewhere in this dissertation, cost is reported as a ratio reflecting the number of development person-hours spent to create one hour of individual instruction.

Table 2 describes the cost of models supporting macroadaptation from six respondents who estimated both development time and instruction time. Table 3 gives the same information for microadaptation, as described by seven respondents. All participants in the study stated that they used microadaptation in their ITSs, and all but one used macroadaptation as well. Macroadaptation costs showed more variation than microadaptation. There was not a significant difference between the cost of developing macroadaptation versus microadaptation (two-tailed T-test).
Certain model types were represented more than once in the responses. Although these responses may come from different participants describing the same project, the likelihood is low because there was no instance when the details from one participant substantially matched another participant’s response.

Table 2: Individual reports of macroadaptation models’ development cost in relation to ITS teaching time.

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlay</td>
<td>24:1</td>
</tr>
<tr>
<td>Decision trees (Classifier)</td>
<td>30:1</td>
</tr>
<tr>
<td>Knowledge tracing</td>
<td>48:1</td>
</tr>
<tr>
<td>Model tracing</td>
<td>100:1</td>
</tr>
<tr>
<td>Overlay</td>
<td>667:1</td>
</tr>
<tr>
<td>Knowledge tracing</td>
<td>1375:1</td>
</tr>
</tbody>
</table>

Table 3: Individual reports of microadaptation models’ development cost in relation to ITS teaching time.

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlay</td>
<td>24:1</td>
</tr>
<tr>
<td>Knowledge tracing</td>
<td>48:1</td>
</tr>
<tr>
<td>Behavior transition networks (Classifier)</td>
<td>50:1</td>
</tr>
<tr>
<td>Differential model (Overlay)</td>
<td>100:1</td>
</tr>
<tr>
<td>Constraint-based model</td>
<td>100:1</td>
</tr>
<tr>
<td>Buggy model</td>
<td>133:1</td>
</tr>
<tr>
<td>Knowledge tracing</td>
<td>450:1</td>
</tr>
</tbody>
</table>

Table 4 shows seven responses relating the cost of building an entire ITS, not just the learner model, to the hours of instruction provided. Each ITS is described by the model types the respondents used. The next section relates the cost of model development to the cost of system development.

In Table 4, two respondents (marked with an asterisk) stated that they used knowledge tracing but did not affirm using model tracing. Since knowledge tracing refers to a way of using a second learner model in conjunction with a cognitive tutor, it may be that these ITSs also used model tracing.
<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classifiers</td>
<td>250:1</td>
</tr>
<tr>
<td>Constraint-based model</td>
<td>333:1</td>
</tr>
<tr>
<td>Overlays</td>
<td>400:1</td>
</tr>
<tr>
<td>Knowledge tracing *</td>
<td>500:1</td>
</tr>
<tr>
<td>Model tracing and differential models</td>
<td>600:1</td>
</tr>
<tr>
<td>Knowledge tracing *</td>
<td>2000:1</td>
</tr>
<tr>
<td>Overlay and buggy models</td>
<td>5333:1</td>
</tr>
</tbody>
</table>

5.1.2.3 Learner model cost as a percentage of ITS cost

Eight participants reported development cost estimates for both a tutoring system as a whole and its learner model. Costs in this section are not ratios, so some new responses can be included that did not appear in the previous section because they lacked instruction time estimates. Taken as an aggregate, these responses show how much of an ITS’s cost goes toward building its learner model.

Responses indicated that, in general, a learner model accounts for about a third of the cost of an ITS, with a mean reported ratio of 33%, a median of 31%, and a standard deviation of 28 percentage points. The responses were overall consistent, so that dropping one low and one high outlier brought the standard deviation to 9 percentage points. The low outlier used an overlay model, and the high outlier used knowledge tracing.

5.1.3 User Model Cost Discussion

5.1.3.1 Interpretations

Although the responses gathered in this survey provide valuable anecdotal insights, there were too few responses to apply more than a gross statistical analysis. However, individual responses suggest some interesting trends. One interesting fact is the
high variability of cost estimates when more than one participant described the same model type. The large differences might be attributable to modeling tasks related to the architecture, such as learning to use a new model type, or unrelated, such as spending more time eliciting knowledge from subject-matter experts. Unfortunately, this study could not determine how much of the variation in cost reports was attributable to the different model types. Further study is needed.

Because there are few reports for each model type, it would be misleading to compare models by their average costs. However, it is still possible to compare the most favorable estimate for each model type. A comparison of best-case scenarios is informative because there is no upper limit on the development effort anyone can expend on any model, but there is a lower limit. Examining the lowest or best reported case provides an “existence proof” to show whether it is at least possible to spend low amounts of time developing a model.

The best-case cost estimates for building a learner model alone cluster into two groups. One group of models has a cost ratio of 50:1 or lower, while the other group has a cost ratio between 100:1 and 133:1. The very high cost estimates in the results are not best-case scenarios because other participants reported lower estimates for the same model categories. The model types in the low-cost group include overlays, classifiers, and knowledge-tracing models (which are typically implemented with a Bayesian or overlay model). The model types that cost more include buggy models, constraint-based models, and the production-rule models in cognitive tutors. Considering best-case
scenarios only, these model types cost between two and 5.5 times as much as the low-cost models.

Table 5: Best-case scenario model costs, as determined by finding the lowest cost ratio reported for each model category.

<table>
<thead>
<tr>
<th>Model Architecture</th>
<th>Cost Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlays</td>
<td>24:1</td>
</tr>
<tr>
<td>Classifiers</td>
<td>30:1</td>
</tr>
<tr>
<td>Knowledge tracing</td>
<td>48:1</td>
</tr>
<tr>
<td>Constraint-based model</td>
<td>100:1</td>
</tr>
<tr>
<td>Production-rule model (model tracing)</td>
<td>100:1</td>
</tr>
<tr>
<td>Buggy model</td>
<td>133:1</td>
</tr>
</tbody>
</table>

Estimating the cost of building a whole ITS, not just a learner model, makes values in this study comparable to published estimates of this figure. The costs of model-tracing tutors and constraint-based tutors reported in this study are approximately equal to figures published in the academic literature.

Using the reasoning discussed above in this section, the two whole-ITS cost ratios over 1000:1 in Table 4 do not represent best-case scenarios because there are lower cost estimates with the same model types. The remaining values in that table are all on the same order of magnitude and even the highest best-case estimate, 600:1 for a model tracing cognitive tutor, was only 2.4 times as high as the lowest estimate. Although these estimates are quite close to each other, the responses do suggest that changing the learner model might halve or double the development time of the entire ITS.

The different responses in Table 4 also suggest an ordering of system development costs by learner model type. Using classifiers as learner models may lead to the fastest ITS development. This confirms intuitions that classifiers, as off-the-shelf tools, are easy to use and do not require publications about their development effort.
Surprisingly, tutoring systems using overlay models fell in the middle of the pack at best, despite the low cost of overlay models compared to other types in this study. However, this unexpected result may be due to the cost of knowledge elicitation on the two projects in question, rather than any costs directly associated with overlay models.

Considering whole-system costs, constraint-based systems are somewhat easier to develop than cognitive tutors, a conclusion which concurs with published anecdotes. The best-case costs of building a tutor with model-tracing or knowledge-tracing are higher than that of a constraint-based tutor, despite the fact that considering the learner model alone, constraints cost the same or more (see Table 5). A possible factor that might contribute to this difference is that constraint-based systems can work with less precise learner models, which might lead to less effort in creating specific hints and remediations for many different errors (Mitrovic, Koedinger, & Martin, 2003). Cognitive tutors, with their model tracing and knowledge tracing algorithms, took the most effort of any ITS to build, confirming the intuition that led to constraint-based modeling and example tracing.

5.1.3.2 Limitations

Limitations of this study include a small population size, possible selection bias, and possible lack of consideration in forming estimates. Although the number of responses reported in this paper is comparable to the number of related publications from the academic community (Section 2.1), that number does not yet reach levels that would allow a detailed statistical analysis. Furthermore, participants were not invited randomly, and invitees with certain characteristics may have been more or less likely to respond.
Finally, ITS researchers who include development costs in publications can support their figures with careful records, while respondents in the present study had to estimate costs after the fact. Because of these limitations, responses in this paper should be viewed as anecdotes rather than predictions of future performance. Although this study presented anecdotal evidence, it is still valuable input into choosing a learner model architecture if the limitations are understood.

5.1.3.3 Study Summary

Learner models often account for one third of ITS development costs. In this study, eleven ITS practitioners from industry, academia, and military organizations shared their valuable experiences to provide anecdotal evidence about those costs.
Table 6: Costs and effects on learning of different model types. Compare to Table 1.

<table>
<thead>
<tr>
<th>Student model</th>
<th>Model detail</th>
<th>Lowest published ratio of development to learning time</th>
<th>Updated cost information the present study gathered</th>
<th>Highest published effect on learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production rules and model tracing</td>
<td>High: all subtasks</td>
<td>200:1</td>
<td>100:1</td>
<td>1.2, compared to classroom learning</td>
</tr>
<tr>
<td>Perturbation and buggy models</td>
<td>High: some or all subtasks</td>
<td>No reports</td>
<td>133:1</td>
<td>Not significant</td>
</tr>
<tr>
<td>Example tracing</td>
<td>Moderate: some subtasks, not all; sometimes tasks</td>
<td>18:1</td>
<td>No reports</td>
<td>0.75, compared to paper homework</td>
</tr>
<tr>
<td>Constraint-based models</td>
<td>Moderate: some or all subtasks or tasks, or a mix</td>
<td>220:1</td>
<td>100:1</td>
<td>1.3, compared to briefing and handout</td>
</tr>
<tr>
<td>Bayesian networks and other classifiers</td>
<td>Low: tasks or skills</td>
<td>No reports</td>
<td>30:1</td>
<td>0.7, compared to the learning task with no hints</td>
</tr>
<tr>
<td>Overlay models</td>
<td>Low: tasks or skills; or some subtasks</td>
<td>No reports</td>
<td>24:1</td>
<td>1.02, compared to on-the-job training</td>
</tr>
</tbody>
</table>

The responses described in this study, which align well with the few published experiences previously available, suggest that certain learner models can be easier to build than others. Overlay models and classifiers used as learner models have the lowest development costs. Constraint-based learner models are approximately as expensive to build as production-rule models. Buggy learner models are the most expensive to develop. The differences in model costs are also reflected in smaller but still noticeable differences in the cost of the entire ITS.

Importantly, this study only addresses development costs. It is likely that more expensive learner models produce such good cognitive fidelity (Neches et al., 1987), effects on learning outcomes, or other benefits that they justify their cost or more.
However, model benefits are both better studied elsewhere in the literature and more specific to individual scenarios than model costs. This survey added to the discourse new knowledge about development costs of real learner models. Practitioners can factor these new data points into ITS planning, while researchers can gain an updated view on the state of the art in modeling and authoring tools.

The survey indicated the need for efficacious models that are possible to create with little effort. It set the stage for research into a new type of learner model that could enhance the easily authored, off-the-shelf classifier group of models with principled planning to increase their effect on learning.

5.2 POMDP ITS Simulation Study

Three experiments were conducted on simulated students to empirically evaluate ITSs driven by different POMDP representations. Examining the performance of different representations helped estimate how the experimental compression schemes might impact real ITS learning outcomes.

5.2.1 Method

POMDPs were developed to tutor a simulated learner on an artificial tutoring problem with carefully controlled characteristics. In three simulation studies, POMDPs and their accompanying policies were presented with simulated students that acted according to the user model $M$ described in Chapter 3. In each simulation, a student was first generated with a random set of knowledge gaps and a random initial cognitive state from among those modeled. An ITS then had a limited number of actions to tutor the
student and remove as many knowledge gaps as possible. Given a student’s hidden mental reality, each ITS action had a specific probability to clear gaps, change cognitive states, and generate new student output observations that let the ITS update its plan. Changes in the students’ mental states always occurred according to the probabilities the user model specified (specific values used appear in the following subsection, 5.2.2). Therefore, $M$ was an exact model of a simulated student, and the simulation experiments tested the experimental POMDPs’ ability to represent $M$ usefully.

The first experiment was designed to explore the effects of problem size on the experimental representation loss during tutoring. Lossless POMDPs were created that used traditional, enumerated state and observation encodings and preserved all information from $M$, perfectly reflecting the simulated students. Experimental POMDPs used either a state queue or observation chains, or both. For problems with $|K| = |Ω| = 1$, the experimental representations do not lose any information. Recall that $K$ represents the knowledge gaps and $Ω$ the observable features in a tutoring problem; the cardinality of these sets determines the sizes of the experimental POMDPs’ state queues and observation chains respectively. As $K$ or $Ω$ grow, the experimental POMDPs compress the information in their states and/or observations, incurring an information loss and possibly a performance degradation.

Since the lossless representation had a perfect model of ground truth, the initial hypothesis in this experiment was that students working with an ITS based on the experimental representations would finish with scores no worse than 0.25 standard deviations above students working with a lossless representation ITS (Glass’ delta). This
threshold was chosen to align with the Institute of Education Science’s standards for identifying substantive effects in educational studies with human subjects (U. S. Department of Education, 2008). A difference of less than 0.25 standard deviations, though it may be statistically significant, is not considered a substantive difference.

The second experiment measured the sensitivity of the experimental representations to the tutoring problem parameters \( a \) and \( d \).

As introduced in Section 3.1, \( d \) is a parameter describing the difficulty of tutoring one concept before another. When \( d \) is low, the concepts may be tutored in any order. Since a state-queue POMDP is constrained in the order it tries to tutor concepts, it might underperform a lossless POMDP on problems where \( d \) is low.

Furthermore, the efficacy of different tutoring actions can vary widely in real-world problems. Efficacy of all actions depends on \( a \), including those not subject to ordering effects.

Simulated tutoring problems with \( a \) in \{0.1, 0.3, 0.5, 0.7, 0.9\} and \( d \) in \{0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.50, 0.75, 1.00\} were evaluated. These problems tested whether performance of the experimental representations depended on properties of the instructional domain that might be difficult to change, such as how much gaps depend on each other and how easy they are for an ITS to address. To maximize performance differences, the largest possible problem size was used, \(|K| = 8\). In problems with more than eight knowledge states, lossless representations were not able to learn useful policies under some parameter values before exhausting all eight gigabytes of available memory.
The experimental representations were hypothesized to perform no more than 0.25 standard deviations worse than the lossless representations. Parameter settings that caused performance to degrade by more than this limit would indicate problem classes for which the experimental representations would be less appropriate in the real world.

The third experiment explored the performance impact of POMDP observations that contain information about more than one gap. Tests on large problems ($|K| \geq 16$) showed the experimental ITSs successfully tutored fewer gaps per turn as size increased. One cause of performance degradation on large problems was conjectured to be the information loss associated with state queuing. As state queues grow longer, the probability increases that a given observation’s limited information describes only states that are not the priority state. If the information loss with large $K$ impacts performance, state queues may be unsuitable for representing some real-world tutoring problems that contain up to a hundred separate concepts (e.g., Payne & Squibb, 1990).

To assess the feasibility of improving large problem performance, the third experiment varied the number of gaps that could be diagnosed based on a single observation. Observations were generated, by the same method used in the other experiments, to describe $n$ equally sized partitions of $K$. When $n = 1$, observations were identical to the previous experiments. With increasing $n$, each observation contained information about the presence or absence of more gaps the ITS should tutor. Observations with high-dimensional information were possible to encode with observation chaining. This experiment could not conclusively demonstrate that low-
information observations degrade state queue performance, but it explored whether performance could be improved in high-information settings.

5.2.2 Experimental Setup

ITS performance in all experiments was evaluated by the number of gaps remaining after $t$ ITS-tutee interactions, including no-op actions. In these experiments, the value $t = |K|$ was chosen to increase differentiation between conditions, giving enough time for a competent ITS to accomplish some tutoring but not to finish in every case.

All gaps had a 50% initial probability of being present. Any combination of gaps was equally likely, except that starting with no gaps was disallowed. Simulated students had a 25% probability of starting in each of the four modeled cognitive states introduced in Section 3.1 and further discussed below, in Section 5.3.

The set of ITS actions included one action to tutor each gap, and a no-op action. Values for $a$ and $d$ were $a_{ij} = 0.5$ where $i = j$, and 0 otherwise; and $d_{ij} = 0.5$ where $i < j$, and 0 otherwise. These values were hypothesized to be moderate enough to avoid extremely good or bad performance, and reasonable for representing at least some real-world tutoring tasks. Nonzero values varied in the second experiment.

POMDP rewards were used only for policy search, not for evaluating POMDP performance. The reward structure was necessarily different for lossless and state-queue POMDPs because state queues do not model some knowledge gap information. However, both reward structures emphasized eliminating gaps as quickly as possible. For lossless POMDPs, any transition from a state with at least one gap to a state with $j$ fewer gaps
earned a reward of $100 j / |K|$. For state-queue POMDPs, any transition that changed the priority gap from a higher priority $i$ to a lower priority $i – j$ earned a reward of $100 j / |K|$, but resolving any gaps other than the priority gap was not rewarded. In both conditions, changes in cognitive state did not earn any reward. The reward discount was 0.90. Goal states were not fully observable.

Policy learning was conducted with the algorithm Successive Approximations of the Reachable Space under Optimal Policies (SARSOP). SARSOP is a point-based learning algorithm that quickly searches large belief spaces by focusing on the points that are more likely to be reached under an approximately optimal policy (Kurniawati et al., 2008). SARSOP’s ability to find approximate solutions to POMDPs with many hidden states makes it suitable for solving ITS problems, especially with the large baseline lossless POMDPs. Version 0.91 of the SARSOP APPL package was used in the present experiment (Wei, 2010).

Training in all conditions consisted of value iteration for 100 trials. In SARSOP, each trial consists of sampling the values at several belief space points along an approximately optimal action sequence, propagating the value estimates up the tree of sampled points, and pruning the tree. Therefore each trial updates multiple belief-space value estimates. Informal tests suggested that learning for more trials did not produce significant performance gains. Evaluations consisted of 10,000 runs for each condition.
5.2.3 Results

The first experiment did not find substantive performance differences between experimental representations and the lossless baseline. Figure 8 shows absolute performance degradation was small in all problems. Degradation did tend to increase with $|K|$. Observation chaining alone did not cause any degradation except in the largest test conducted, when $|K| = 8$. State queues, alone or with observation chains, degraded performance by less than 10% of a standard deviation for all values of $|K|$.

Lossless POMDPs were not able to represent problems with $|K| > 8$. However, extrapolating from the results of the first experiment suggested that if a lossless POMDP with larger memory and processor resources could represent a problem with $|K| = 16$, its performance would be more than 0.25 standard deviations better than the experimental representations. This extrapolation motivated the third experiment, which tested the extent to which more informative observations could mitigate performance degradation on large problems.
Figure 8: Direct comparison to a lossless POMDP on problems with $|K| \leq 8$, $a = 0.5$, $d = 0.5$ shows the experimental representations did not substantively degrade tutoring performance.

The second experiment tested for performance differences under extreme values of $a$ and $d$. In lossless POMDPs, performance depended mostly on action efficacy. Varying the strength of the priority effect caused relatively small performance differences. Furthermore, observation chaining POMDPs performed substantively the same as lossless POMDPs under every combination of parameters.

Differences did appear for POMDPs with a state queue alone or a state queue combined with observation chaining. With a state queue alone, POMDPs performed substantively worse than the lossless baseline on tutoring problems with $d < 0.25$. The relative degradation increased on problems with greater action efficacy because the lossless POMDPs’ better performance magnified small absolute differences. With a state
queue and observation chaining combined, POMDPs performed substantively worse on problems with $d < 0.20$. The improvement over queues alone may be attributable to more efficient policy learning possible with smaller $\Omega$. 
Figure 9 and Table 7 show the performance degradation in the second experiment POMDP using both state queues and observation chains, as a percentage of the lossless performance. Performance changes that were not substantive, but were close to random noise, ranged from 0% to 50% worse than the lossless representation. The first substantive difference came at $d = 0.15$, with a 53% degradation. The most substantive difference was 179% worse, but represented an absolute difference of only 0.22 (an average of 0.34 gaps remaining, compared to 0.12 with a lossless POMDP).

![Figure 9: When $|K| = 8$, $|A| = 8$, and $|\Omega| = 64$, as here, lossless POMDP performance (a) varies with action efficacy, not priority effect. Experimental POMDPs perform about the same as lossless in many conditions (b, c, d), but state queues substantively degrade performance at some points (circled), indicating they are unsuitable for priority effect $d < 0.25$.](image-url)
Table 7: Performance of compression schemes as compared to a lossless representation. Figure 9 visualizes this data.

<table>
<thead>
<tr>
<th>d</th>
<th>a</th>
<th>Lossless (σ)</th>
<th>State Queue (δ)</th>
<th>Obs. Chain (δ)</th>
<th>Queue + Chain (δ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.10</td>
<td>2.99 (1.59)</td>
<td>3.24 (0.16)</td>
<td>2.98 (0.00)</td>
<td>3.23 (0.15)</td>
</tr>
<tr>
<td>0.00</td>
<td>0.30</td>
<td>1.55 (1.43)</td>
<td>2.20 (0.45)</td>
<td>1.57 (0.01)</td>
<td>2.08 (0.37)</td>
</tr>
<tr>
<td>0.00</td>
<td>0.50</td>
<td>0.75 (1.06)</td>
<td>1.40 (0.61)</td>
<td>0.76 (0.01)</td>
<td>1.24 (0.47)</td>
</tr>
<tr>
<td>0.00</td>
<td>0.70</td>
<td>0.32 (0.69)</td>
<td>0.82 (0.71)</td>
<td>0.32 (0.00)</td>
<td>0.67 (0.50)</td>
</tr>
<tr>
<td>0.00</td>
<td>0.90</td>
<td>0.12 (0.40)</td>
<td>0.41 (0.71)</td>
<td>0.12 (-0.01)</td>
<td>0.34 (0.54)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.10</td>
<td>3.06 (1.59)</td>
<td>3.25 (0.12)</td>
<td>3.08 (0.01)</td>
<td>3.21 (0.10)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.30</td>
<td>1.66 (1.49)</td>
<td>2.20 (0.36)</td>
<td>1.64 (-0.01)</td>
<td>2.12 (0.31)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.50</td>
<td>0.82 (1.13)</td>
<td>1.39 (0.51)</td>
<td>0.83 (0.01)</td>
<td>1.27 (0.40)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.70</td>
<td>0.37 (0.75)</td>
<td>0.83 (0.61)</td>
<td>0.37 (0.00)</td>
<td>0.69 (0.42)</td>
</tr>
<tr>
<td>0.05</td>
<td>0.90</td>
<td>0.16 (0.47)</td>
<td>0.42 (0.55)</td>
<td>0.15 (-0.01)</td>
<td>0.36 (0.42)</td>
</tr>
<tr>
<td>0.10</td>
<td>0.10</td>
<td>3.10 (1.60)</td>
<td>3.24 (0.09)</td>
<td>3.11 (0.00)</td>
<td>3.23 (0.08)</td>
</tr>
<tr>
<td>0.10</td>
<td>0.30</td>
<td>1.76 (1.56)</td>
<td>2.23 (0.30)</td>
<td>1.75 (-0.01)</td>
<td>2.13 (0.24)</td>
</tr>
<tr>
<td>0.10</td>
<td>0.50</td>
<td>0.91 (1.21)</td>
<td>1.46 (0.46)</td>
<td>0.91 (0.00)</td>
<td>1.30 (0.33)</td>
</tr>
<tr>
<td>0.10</td>
<td>0.70</td>
<td>0.44 (0.84)</td>
<td>0.86 (0.5)</td>
<td>0.44 (0.01)</td>
<td>0.72 (0.34)</td>
</tr>
<tr>
<td>0.10</td>
<td>0.90</td>
<td>0.19 (0.55)</td>
<td>0.43 (0.43)</td>
<td>0.19 (0.00)</td>
<td>0.36 (0.32)</td>
</tr>
<tr>
<td>0.15</td>
<td>0.10</td>
<td>3.15 (1.62)</td>
<td>3.27 (0.07)</td>
<td>3.13 (-0.01)</td>
<td>3.23 (0.06)</td>
</tr>
<tr>
<td>0.15</td>
<td>0.30</td>
<td>1.81 (1.59)</td>
<td>2.22 (0.26)</td>
<td>1.83 (0.01)</td>
<td>2.16 (0.22)</td>
</tr>
<tr>
<td>0.15</td>
<td>0.50</td>
<td>1.01 (1.30)</td>
<td>1.48 (0.36)</td>
<td>1.01 (0.00)</td>
<td>1.33 (0.25)</td>
</tr>
<tr>
<td>0.15</td>
<td>0.70</td>
<td>0.50 (0.92)</td>
<td>0.86 (0.40)</td>
<td>0.48 (-0.02)</td>
<td>0.76 (0.29)</td>
</tr>
<tr>
<td>0.15</td>
<td>0.90</td>
<td>0.23 (0.63)</td>
<td>0.43 (0.32)</td>
<td>0.23 (0.00)</td>
<td>0.37 (0.22)</td>
</tr>
<tr>
<td>0.20</td>
<td>0.10</td>
<td>3.16 (1.60)</td>
<td>3.27 (0.07)</td>
<td>3.15 (-0.01)</td>
<td>3.26 (0.06)</td>
</tr>
<tr>
<td>0.20</td>
<td>0.30</td>
<td>1.91 (1.62)</td>
<td>2.23 (0.20)</td>
<td>1.95 (0.02)</td>
<td>2.16 (0.16)</td>
</tr>
<tr>
<td>0.20</td>
<td>0.50</td>
<td>1.08 (1.37)</td>
<td>1.51 (0.31)</td>
<td>1.10 (0.01)</td>
<td>1.35 (0.20)</td>
</tr>
<tr>
<td>0.20</td>
<td>0.70</td>
<td>0.57 (1.00)</td>
<td>0.88 (0.30)</td>
<td>0.57 (-0.01)</td>
<td>0.79 (0.21)</td>
</tr>
<tr>
<td>0.20</td>
<td>0.90</td>
<td>0.27 (0.69)</td>
<td>0.45 (0.26)</td>
<td>0.28 (0.01)</td>
<td>0.41 (0.20)</td>
</tr>
<tr>
<td>0.25</td>
<td>0.10</td>
<td>3.19 (1.59)</td>
<td>3.28 (0.06)</td>
<td>3.21 (0.01)</td>
<td>3.28 (0.06)</td>
</tr>
<tr>
<td>0.25</td>
<td>0.30</td>
<td>2.02 (1.65)</td>
<td>2.25 (0.14)</td>
<td>2.01 (-0.01)</td>
<td>2.16 (0.09)</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50</td>
<td>1.18 (1.41)</td>
<td>1.47 (0.21)</td>
<td>1.21 (0.02)</td>
<td>1.39 (0.15)</td>
</tr>
<tr>
<td>0.25</td>
<td>0.70</td>
<td>0.63 (1.06)</td>
<td>0.89 (0.25)</td>
<td>0.65 (0.02)</td>
<td>0.82 (0.18)</td>
</tr>
<tr>
<td>0.25</td>
<td>0.90</td>
<td>0.31 (0.75)</td>
<td>0.48 (0.21)</td>
<td>0.33 (0.02)</td>
<td>0.42 (0.14)</td>
</tr>
<tr>
<td>0.50</td>
<td>0.10</td>
<td>3.29 (1.52)</td>
<td>3.31 (0.01)</td>
<td>3.3 (0.00)</td>
<td>3.31 (0.01)</td>
</tr>
<tr>
<td>0.50</td>
<td>0.30</td>
<td>2.29 (1.63)</td>
<td>2.36 (0.05)</td>
<td>2.25 (-0.02)</td>
<td>2.33 (0.02)</td>
</tr>
<tr>
<td>0.50</td>
<td>0.50</td>
<td>1.47 (1.52)</td>
<td>1.61 (0.09)</td>
<td>1.48 (0.00)</td>
<td>1.59 (0.07)</td>
</tr>
<tr>
<td>0.50</td>
<td>0.70</td>
<td>0.85 (1.25)</td>
<td>1.02 (0.14)</td>
<td>0.88 (0.02)</td>
<td>0.97 (0.10)</td>
</tr>
<tr>
<td>0.50</td>
<td>0.90</td>
<td>0.43 (0.92)</td>
<td>0.55 (0.13)</td>
<td>0.46 (0.04)</td>
<td>0.53 (0.11)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.10</td>
<td>3.34 (1.49)</td>
<td>3.38 (0.03)</td>
<td>3.34 (0.00)</td>
<td>3.38 (0.02)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.30</td>
<td>2.40 (1.57)</td>
<td>2.42 (0.01)</td>
<td>2.39 (-0.01)</td>
<td>2.44 (0.02)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.50</td>
<td>1.65 (1.53)</td>
<td>1.75 (0.07)</td>
<td>1.64 (0.00)</td>
<td>1.76 (0.08)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.70</td>
<td>1.04 (1.38)</td>
<td>1.15 (0.08)</td>
<td>1.01 (-0.02)</td>
<td>1.18 (0.10)</td>
</tr>
<tr>
<td>0.75</td>
<td>0.90</td>
<td>0.57 (1.07)</td>
<td>0.66 (0.09)</td>
<td>0.59 (0.02)</td>
<td>0.67 (0.10)</td>
</tr>
<tr>
<td>1.00</td>
<td>0.10</td>
<td>3.39 (1.49)</td>
<td>3.39 (0.00)</td>
<td>3.40 (0.01)</td>
<td>3.38 (-0.01)</td>
</tr>
</tbody>
</table>
The third experiment (Figure 10) showed the effect on tutoring performance of observations with more information. For $|K| = 32$, tutoring left an average of 9.4 gaps when each observation had information about one gap, and 3.7 when each observation described 16 gaps. For $|K| = 64$, the number of gaps remaining went from 21.5 to 9.2. Finally, when $|K| = 128$ performance improved from leaving 47.9 gaps to leaving just 22.3 with information about 32 gaps in each observation. However, for this size problem, doubling the available information again did not result in further performance improvement.
Figure 10: Performance on large problems can be improved up to approximately double, if every observation contains information about multiple features. Observation chaining greatly increases the number of independent dimensions a single observation can completely describe (the number above each column).

5.2.4 Simulation Study Discussion and Summary

Together, the simulated experiments in this section show encouraging results for the practicality of POMDP ITSs.

The first experiment demonstrated that information compression in the experimental representations did not lead to substantively worse performance for problems small enough to compare. The size limit on lossless representations ($|K| \leq 8$) is too restrictive for many ITS applications. The limit was caused by memory requirements of policy iteration. Parameters chosen to shorten training runs could increase maximum problem size by only one or two more. Furthermore, one performance advantage of using
enumerated representations stems from their ability to encode complex relationships between states or observations. But the improvement may not be useful if it is impractical to learn or specify each of the thousands of relationships.

The second experiment suggested that although certain types of problems are inappropriate to encode with the experimental representations, problems with certain characteristics are probably suitable. Whenever gap priority had an effect on the probability of clearing gaps $d \geq 0.2$, state queues could discard large amounts of information (as they do) and retain substantively the same final outcomes. Informal discussions with subject-matter experts suggest many real-world tutoring problems have priorities near $d \approx 0.5$.

Although it would be premature to compare simulation results to the performance of any ITS on real learners, there is a way to provide context for the value of a POMDP in tutoring the simulated problems. Figure 11 compares the results of the second experiment to a simulated reactive ITS that does not model knowledge gaps, but simply tutors the last gap observed. The comparison shows that unlike POMDPs, the reactive ITS’s performance was affected by the priority effect $d$. Figure 11b displays the gaps the POMDP ITS left remaining as a percentage of the reactive ITS’s gap counts. The POMDP performed substantively better in almost all cases, achieving learning gains between 0.09 and 0.72 standard deviations better.
Figure 11: Simply tutoring the most recently observed gap (a) causes poor performance when gap priority is important. POMDP scores (see Figure 9a) are a fraction of the comparison scores at all points (b). The POMDP performs substantively better at many points (circled).

The third experiment demonstrated one method to address performance degradation in large problems. Adding more assessment information to each observation empirically improved performance. Observing many dimensions at once is practical for a subset of real-world ITS problems. For example, it may be realistic to assess any tutee performance, good or bad, as demonstrating a grasp of all fundamental skills that lead up to that point. In such a case one observation could realistically rule out many gaps.

In the third experiment, problems with $|K| = 128$ and more than 32 gaps described in each observation did not finish policy learning before reaching memory limits. This suggests an approximate upper limit on problem size for the experimental representations. POMDPs that are able to encompass 128 states could be sufficient to control ITSs that tutor many misconceptions, such as (Payne & Squibb, 1990). Even larger material could be factored to fit within the limit by dividing it into chapters or tutoring sessions.

The results in this section are limited because they apply only to the performance of the compressed representations in combination with the SARSOP sampling and value iteration algorithm. Other solvers might perform differently.
In summary, the three simulation experiments suggest that POMDPs compressed with state queues and observation chains are promising for use on ITS problems when topics must be taught in order. The CFF domain and other domains where model structure reflects instructional material ordering and dependencies (e.g., Conati et al., 2002; Luckin & du Boulay, 1999) align well with POMDP ITSs. The structures may be less appropriate when topics can appear in many orders, such as in exploratory learning environments (e.g., Shute & Glaser, 1990). Second, although state queues were shown in the present work to accommodate state counts that can model many real ITS problems, they are still not appropriate for very large numbers of states, such as in highly detailed moment-to-moment models like production rule systems use (e.g., J. R. Anderson, 1993). Finally, observation chains are helpful when a large observation space can be factored into several conditionally independent observations, for example by an assessment module. However, observation chains are not useful if a large number of observations must relate directly to states through a complex function that cannot be factored. In such cases, a different representation must be used.

Although the compressed representations are not suitable for some tutoring problems, they can represent a wide range of problems without damaging instructional efficacy.

5.3 Inquiry Modeling Observational Study

An observational study of trainees in a real-world training scenario was conducted. Trainers and trainees participating in the study were human, rather than
simulated. In general, the study characterized the number, form, and semantic content of trainee questions. In relation to the specific training task, the study helped select members of $K$, expected misconceptions or knowledge gaps, and $C$, expected cognitive states that affect learning or performance. The study provided support for the hypothesis that user questions in an ITS can act as valuable user model inputs.

5.3.1 Assessment and Diagnosis

5.3.1.1 Underlying States and Gaps

Several cognitive states were hypothesized to affect performance and learning. The experiment gathered information on participants’ cognitive load, affective states, and various gaps or specific reasons for missing or incorrect knowledge or performance.

Cognitive load, which in the context of this experiment is a broad term representing instantaneous loads on working memory, selective attention, and mental processing capacity, is known to affect performance and learning (Sweller, 1988). The cognitive load participants encountered was measured with self-reports of mental effort, a component of the holistic load construct which has been shown to correlate with overall cognitive load (Paas, 1992). Table 8 lists the levels of cognitive load (mental effort) participants could report.
Table 8: Levels of cognitive load trainees might experience during training or performance.

<table>
<thead>
<tr>
<th>Report</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Very, very low mental effort</td>
</tr>
<tr>
<td>2</td>
<td>Very low mental effort</td>
</tr>
<tr>
<td>3</td>
<td>Low mental effort</td>
</tr>
<tr>
<td>4</td>
<td>Rather low mental effort</td>
</tr>
<tr>
<td>5</td>
<td>Neither low nor high mental effort</td>
</tr>
<tr>
<td>6</td>
<td>Rather high mental effort</td>
</tr>
<tr>
<td>7</td>
<td>High mental effort</td>
</tr>
<tr>
<td>8</td>
<td>Very high mental effort</td>
</tr>
<tr>
<td>9</td>
<td>Very, very high mental effort</td>
</tr>
</tbody>
</table>

Affective states are a subset of cognitive states describing emotion or mood. Affect can influence how learners view and interact with a tutor and with the material. Learner affect has been studied in computer learning environments such as Prime Climb (Conati, 2002; Conati & Maclaren, 2005, 2009), Crystal Island (S. McQuiggen, Robison, & Lester, 2008; Robison et al., 2009), The Incredible Machine (Baker et al., 2007; Rodrigo et al., 2007), and AutoTutor (Craig et al., 2004; D’Mello, Person, & Lehman, 2009; D’Mello et al., 2007). The particular affective states investigated in this study were the same ones previously found to influence performance and instructional effectiveness by Craig and others (2004), Baker and others (2007), Rodrigo and others (2007), and D’Mello and others (2007).

Table 9 lists the six affective states that were studied. The functional definitions for these states are listed as they were presented to study participants and are based on those used in previous research on these states (D’Mello, Craig, Sullins, & Graesser, 2006). In accordance with the several studies on these affective states, the first four states named were hypothesized to appear more often during training than the last two.
Based on input from subject-matter experts and pilot studies, 31 knowledge states were initially hypothesized to comprise a sufficient model of the misconceptions and gaps trainees could have. Table 10 lists the required knowledge or skills and corresponding gaps or misconceptions for these 31 states.

<table>
<thead>
<tr>
<th>State</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boredom</td>
<td>being weary or restless through lack of interest</td>
</tr>
<tr>
<td>Confusion</td>
<td>a noticeable lack of understanding</td>
</tr>
<tr>
<td>Flow</td>
<td>smooth and uninterrupted progress, a state of interest that results from involvement in an activity</td>
</tr>
<tr>
<td>Frustration</td>
<td>dissatisfied or annoyed</td>
</tr>
<tr>
<td>Delight</td>
<td>a high degree of satisfaction</td>
</tr>
<tr>
<td>Surprise</td>
<td>wonder or amazement, especially from the unexpected</td>
</tr>
</tbody>
</table>

Table 9: Affective states trainees might experience during scenarios, with the definitions presented to study participants.
Table 10: KSAs required for good performance in the simplified CFF domain.

<table>
<thead>
<tr>
<th>KSA Type</th>
<th>Correct KSA</th>
<th>Corresponding gaps</th>
</tr>
</thead>
</table>
| Domain KSAs | Correctly identifies friend or foe | Believes T72 is a foe  
Believes ZSU is a foe  
Believes BMP is a friend  
Believes BTR-70 is a friend |
| Knows general steps to call for fire | Does not know one or more general steps to call for fire |
| Issues correct warning order | Targets moving vehicles with Adjust Fire  
Targets stationary vehicles with Fire for Effect |
| Directs correct method of engagement | Targets tanks with HE/Quick rounds  
Targets light vehicles with ICM rounds |
| Targets highest-priority enemies first | Targets distant enemies before closer alternatives  
Targets stationary vehicles before closer alternatives  
Believes distance determines priority more strongly than vehicle movement  
Believes vehicle type helps determine priority |
| Simulator KSAs | Uses GPS to determine own location | Cannot activate GPS or read GPS correctly |
| Uses rangefinder to identify distance and bearing | Cannot activate rangefinder or read bearings correctly  
Reads distances incorrectly |
| Uses radio to call for fire | Cannot activate radio  
Cannot transmit a line in the radio menu  
Cannot fill in radio menu posrep line  
Cannot fill in radio menu warning order  
Cannot fill in radio menu location method  
Confused about radio menu target U/D fields (do not need to be changed)  
Cannot fill in radio menu target description  
Cannot fill in radio menu method of engagement  
Cannot fill in radio menu method of control  
Confused about radio menu target dimension fields (do not need to be changed) |
<table>
<thead>
<tr>
<th>KSA Type</th>
<th>Correct KSA</th>
<th>Corresponding gaps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Cannot acknowledge MTO in the radio menu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Confused about radio menu MTO fields (do not need to be changed)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cannot fill in radio menu end of mission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Confused about radio menu end of mission fields (do not need to be changed)</td>
</tr>
<tr>
<td>Perceptual KSAs</td>
<td>Sees enemy vehicles</td>
<td>Does not see some enemy vehicle x</td>
</tr>
</tbody>
</table>

5.3.1.2 Performance Assessment

Participants’ performances were aggregated from simulator logs and written tests. Assessing performance allowed exploration of how the types and frequency of learner questions or other training interaction related to learning.

Final performance was used as a proxy for learning because, based on intake forms, all trainees initially had no knowledge of the CFF task. It was not possible to administer a pretest for the same reason, and in addition pretests in this domain can act as advance organizers, unduly influencing training outcomes.

Performance in the simulator was measured in time needed to generate a CFF, number of CFFs that resulted in a hit, errors in CFF target descriptions or details, and target prioritization. Performance on written tests was assessed according to a scoring rubric.

5.3.1.3 Help-Seeking Classification

During training, participants were either allowed or encouraged (depending on experimental condition) to ask for help from a human trainer. Any explicit or implicit
help requests were categorized according to several features of their syntax and content.
Understanding the kinds of questions learners ask during training can help with designing
an ITS for the CFF domain specifically and also with designing inquiry-modeling ITSs in
general.

Each help-seeking event was classified in eight dimensions: grammatical mood,
specification, trigger, form, knowledge type, cognitive process, domain topic, and answer
elicited. Table 11 lists the possible values for each feature.

Grammatical mood and specification (Graesser, 1992) are surface features of
help-seeking events. Learning about them may help in the future with accepting natural-
language help requests or improving help request recognition. Mood indicates whether a
help request was phrased as a question, a statement (“I don’t get this”), or a command
(“Tell me again how this works”). Specification is a measure of the extent to which
context, like knowledge of what is happening in the simulator or previous conversational
turns, is needed to understand the help request.

The trigger (Graesser, 1992; Graesser, Person, & Huber, 1992) of a help-seeking
event records what caused the learner to ask for help and what the broad goal of the help
request was. The event’s form (Graesser, 1992; Graesser, Person, & Huber, 1993) refers
to a finer-grained classification of the knowledge sought. Form is related to the
knowledge type and cognitive process described by the modified Bloom taxonomy (L.
W. Anderson & Krathwohl, 2001), but may be easier to determine automatically based on
syntactical features. The Bloom taxonomy, in turn, is well known to educators and
describes the depth of the knowledge requested.
The domain topic of a help request describes what part of the CFF the learner asked about. Categorizing the correctness of the answer the help request elicited could help explain the learner’s behavior after the help-seeking event, such as asking another question, correcting a misconception, or continuing as before.

Table 11: Help seeking event taxonomy.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammatical mood</td>
<td>Interrogative</td>
</tr>
<tr>
<td></td>
<td>Declarative</td>
</tr>
<tr>
<td></td>
<td>Imperative</td>
</tr>
<tr>
<td>Specification</td>
<td>High context included</td>
</tr>
<tr>
<td></td>
<td>Some context</td>
</tr>
<tr>
<td></td>
<td>Almost no context</td>
</tr>
<tr>
<td>Trigger</td>
<td>Identified missing or forgotten information</td>
</tr>
<tr>
<td></td>
<td>Identified an apparent contradiction or misconception</td>
</tr>
<tr>
<td></td>
<td>Establishing common ground such as history or definitions</td>
</tr>
<tr>
<td></td>
<td>Coordinating group actions or conversational control</td>
</tr>
<tr>
<td>Form</td>
<td>Verification</td>
</tr>
<tr>
<td></td>
<td>Disjunctive</td>
</tr>
<tr>
<td></td>
<td>Concept completion</td>
</tr>
<tr>
<td></td>
<td>Feature specification</td>
</tr>
<tr>
<td></td>
<td>Quantification</td>
</tr>
<tr>
<td></td>
<td>Definition</td>
</tr>
<tr>
<td></td>
<td>Example</td>
</tr>
<tr>
<td></td>
<td>Comparison</td>
</tr>
<tr>
<td></td>
<td>Interpretation</td>
</tr>
<tr>
<td></td>
<td>Causal antecedent</td>
</tr>
<tr>
<td></td>
<td>Causal consequence</td>
</tr>
<tr>
<td></td>
<td>Goal orientation</td>
</tr>
<tr>
<td></td>
<td>Procedural</td>
</tr>
<tr>
<td></td>
<td>Enablement</td>
</tr>
<tr>
<td></td>
<td>Expectational</td>
</tr>
<tr>
<td></td>
<td>Judgmental</td>
</tr>
<tr>
<td></td>
<td>Assertion</td>
</tr>
<tr>
<td>Knowledge type</td>
<td>Factual</td>
</tr>
<tr>
<td></td>
<td>Conceptual</td>
</tr>
<tr>
<td></td>
<td>Procedural</td>
</tr>
<tr>
<td></td>
<td>Metacognitive</td>
</tr>
<tr>
<td><strong>Feature</strong></td>
<td><strong>Possible Values</strong></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cognitive process</td>
<td>Remember</td>
</tr>
<tr>
<td></td>
<td>Understand</td>
</tr>
<tr>
<td></td>
<td>Apply</td>
</tr>
<tr>
<td></td>
<td>Analyze / Evaluate / Create</td>
</tr>
<tr>
<td>Timing</td>
<td>Slides 1</td>
</tr>
<tr>
<td></td>
<td>Practice 1</td>
</tr>
<tr>
<td></td>
<td>Slides 2</td>
</tr>
<tr>
<td></td>
<td>Practice 2</td>
</tr>
<tr>
<td></td>
<td>Test 1</td>
</tr>
<tr>
<td></td>
<td>Test 2</td>
</tr>
<tr>
<td></td>
<td>Written Test</td>
</tr>
<tr>
<td>Domain topic</td>
<td>General CFF steps</td>
</tr>
<tr>
<td></td>
<td>Simulator, interface, or tool usage</td>
</tr>
<tr>
<td></td>
<td>Identify friend or foe</td>
</tr>
<tr>
<td></td>
<td>How to engage</td>
</tr>
<tr>
<td></td>
<td>Target prioritization</td>
</tr>
<tr>
<td></td>
<td>Other on-topic</td>
</tr>
<tr>
<td></td>
<td>Off-topic</td>
</tr>
<tr>
<td>Answered correctly</td>
<td>Not at all</td>
</tr>
<tr>
<td></td>
<td>Partially</td>
</tr>
<tr>
<td></td>
<td>Completely</td>
</tr>
<tr>
<td></td>
<td>With elaboration</td>
</tr>
</tbody>
</table>

5.3.2 Method

Participants \((n = 14)\) were recruited from among psychology undergraduates to participate in this study. Participants were required to be United States citizens. Ten participants were male and four were female. Participants’ average age was 18.9 \((\sigma = 1.4)\). No participants had any ROTC experience or any prior knowledge about the CFF domain. All but one of the participants felt at least average comfort with using a computer.

Participants performed CFF tasks in four FOPCSIM scenarios and completed written knowledge tests about the simulator and task domain. For transcription and
encoding purposes, video and audio records were made during the study. A video camera captured the DVTE screen contents while recording audio of the participant’s utterances. Using the recordings, participants’ utterances and performance were coded after the session according to the assessment taxonomy described above.

On arrival, participants were randomly assigned to one of group A or group B. The different groups completed the same tasks, but were observed via different methods.

Participants in group A (n = 7) employed a concurrent think-aloud technique (Ericsson & Simon, 1984; Fonteyn, Kuipers, & Grobe, 1993). While interacting with the DVTE, participants described what they were thinking about, including simulated events they noticed, problems that arose, and how they tried to solve those problems. An experimenter was present while the participant used the DVTE, and in some cases prompted the participant to talk about specific thoughts according to pre-arranged triggers. Participants in the think-aloud condition were not instructed to ask the experimenter for help during training, although they were allowed to do so.

Participants in group B (n = 7) did not use a think-aloud method, but were encouraged to interact with the experimenter by asking for expert assistance such as reminders, clarifications, or hints. Participants were reminded once at the start of each simulator session to ask the investigator for help as needed. Participants’ requests for assistance were expected to reveal some subset of the same information made available through think-aloud in group A.

Each training and testing session took 2.5 hours. The agenda in Table 12 describes the specific scenarios and tests each participant carried out.
Table 12: Observational study agenda.

<table>
<thead>
<tr>
<th>Agenda Task</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2 minutes</td>
</tr>
<tr>
<td>Consent and background forms</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Procedural introduction and practice**</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Simulator introduction</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Practice 1, Simulator practice*</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Domain task introduction</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Practice 2, Domain task practice*</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Domain written knowledge test</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Test 1, Domain task test*</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Domain transfer task introduction</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Test 2, Domain transfer task test*</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Domain transfer written test</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Debrief</td>
<td>3 minutes</td>
</tr>
</tbody>
</table>

Agenda lines marked with one asterisk took place in the DVTE, and participants’ experiences differed between groups A and B. During these times, group A practiced think-aloud while group B asked questions of the experimenter. The agenda line marked with two asterisks also differed according to group. This line represents a brief introduction and practice of either how to use the think-aloud procedure or how to ask useful questions that can improve learning.

Practice 1 introduced how to use the simulator and the simulated tools. There were only two potential targets and both were enemies. Practice 2 introduced friend or foe differentiation, the two firing patterns to choose between, the two types of munitions to use, and the two reasons for prioritizing one enemy target over another. Practice 1 and Practice 2 were each preceded by an instructional presentation. Test 1 tested the same domain task as Practice 2, while Test 2 required the same steps as the domain task but with the designated defense position moved away from the trainee’s perspective and with
additional pressure from an on-screen count of friendly casualties. Written questions further tested participants’ memory and deep understanding of the instructional material.

In previous studies using the same CFF task, Practice 2 was normally broken into two or more scenarios (Vogel-Walcutt, Fiore, Bowers, & Nicholson, 2009; Vogel-Walcutt et al., 2008). The present study combined these scenarios to introduce more material at once, for the purpose of degrading trainees’ support and causing them to display more misconceptions. Furthermore, some facts participants needed to know for peak performance were not mentioned during teaching. For example, participants were taught to prioritize moving targets and to prioritize close targets. They were not told how to combine these instructions—correct prioritization requires engaging all moving targets from closest to furthest before engaging even the closest static target. Participants needed to recognize that they did not have enough information and ask questions during training, or they would perform poorly. Trainers did not volunteer corrections except when needed to let training continue. Trainees could recognize a need for more information during normal simulator interactions, during written tests, or when training environment messages appeared. (Training environment messages were text overlays that obscured one corner of the screen with a correcting message after some incorrect CFFs.)

The cognitive and knowledge states trainees displayed during training were observed and recorded by the trainer and a second observer. Possible observation sources included participants’ comments or think-aloud utterances, questions posed to the trainer, and performance in the simulator. Each observation was coded as evidence for or against the presence of the hypothesized knowledge and cognitive states. Together, these
observations approximate what an expert human trainer could hope to know about a trainee. An ITS user model would have access to only a subset of these observations.

After each simulator interaction, trainees reported how often they had felt each of six affective states on a seven-point Likert scale. Each scale was labeled with ratings from “Never” through “Half the time” to “Constantly.” This affective-state measure was devised for the present study. The measure was presented to participants after each interaction with the simulator.

Retrospective trainee reports of mental effort were collected after each simulator and non-simulator task. A cognitive load questionnaire (CLQ) asked participants to report their mental effort during each preceding task. Although subjective and retrospective, the CLQ is a simple measure for participants to complete, and responses have been shown to correlate with cognitive load (Paas, 1992). The CLQ was presented after each time a participant viewed instructional slides, used the simulator, or answered written questions.

After all teaching, training, and testing, participants completed an exit questionnaire with Likert-scale responses describing their perceptions of the experience. They reported how they felt about the instruction and what effects they believed the question-asking or think-aloud interactions had on their learning and performance. The exit questionnaire was devised for the present study.
5.3.3 Results and Discussion

5.3.3.1 Help Request Characteristics

A total of 414 help requests were recorded and coded according to the taxonomy of help requests. These results explore the nature of questions learners ask during training in a virtual environment. The feature distributions (Table 13) were similar to those Graesser and Person reported (1994) in their study of help requests during tutoring of algebra and research methods material.

Table 13: Help request feature observation frequency.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Possible Values</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grammatical mood</td>
<td>Interrogative</td>
<td>344</td>
<td>83%</td>
</tr>
<tr>
<td></td>
<td>Declarative</td>
<td>63</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>Imperative</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td>Specification</td>
<td>High context included</td>
<td>29</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>Some context</td>
<td>370</td>
<td>89%</td>
</tr>
<tr>
<td></td>
<td>Almost no context</td>
<td>12</td>
<td>3%</td>
</tr>
<tr>
<td>Trigger</td>
<td>Identified missing or forgotten information</td>
<td>277</td>
<td>67%</td>
</tr>
<tr>
<td></td>
<td>Identified an apparent contradiction or misconception</td>
<td>63</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>Establishing common ground such as history or definitions</td>
<td>56</td>
<td>14%</td>
</tr>
<tr>
<td></td>
<td>Coordinating group actions or conversational control</td>
<td>15</td>
<td>4%</td>
</tr>
<tr>
<td>Feature</td>
<td>Possible Values</td>
<td>Count</td>
<td>Percentage</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>Form</td>
<td>Verification</td>
<td>142</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>Disjunctive</td>
<td>43</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Concept completion</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Feature specification</td>
<td>54</td>
<td>13%</td>
</tr>
<tr>
<td></td>
<td>Quantification</td>
<td>5</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Definition</td>
<td>39</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>Example</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Comparison</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Interpretation</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Causal antecedent</td>
<td>27</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>Causal consequence</td>
<td>10</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Goal orientation</td>
<td>6</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Procedural</td>
<td>66</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>Enablement</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Expectational</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Judgmental</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Assertion</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td>Knowledge type</td>
<td>Factual</td>
<td>213</td>
<td>51%</td>
</tr>
<tr>
<td></td>
<td>Conceptual</td>
<td>43</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Procedural</td>
<td>152</td>
<td>37%</td>
</tr>
<tr>
<td></td>
<td>Metacognitive</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>Cognitive process</td>
<td>Remember</td>
<td>119</td>
<td>29%</td>
</tr>
<tr>
<td></td>
<td>Understand</td>
<td>196</td>
<td>47%</td>
</tr>
<tr>
<td></td>
<td>Apply</td>
<td>92</td>
<td>22%</td>
</tr>
<tr>
<td></td>
<td>Analyze / Evaluate / Create</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td>Timing</td>
<td>Slides 1</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Practice 1</td>
<td>202</td>
<td>49%</td>
</tr>
<tr>
<td></td>
<td>Slides 2</td>
<td>6</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Practice 2</td>
<td>134</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td>Test 1</td>
<td>34</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>Test 2</td>
<td>31</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>Written Test</td>
<td>3</td>
<td>1%</td>
</tr>
<tr>
<td>Domain topic</td>
<td>General CFF steps</td>
<td>127</td>
<td>31%</td>
</tr>
<tr>
<td></td>
<td>Simulator, interface, or tool usage</td>
<td>99</td>
<td>24%</td>
</tr>
<tr>
<td></td>
<td>Identify friend or foe</td>
<td>79</td>
<td>19%</td>
</tr>
<tr>
<td></td>
<td>How to engage</td>
<td>73</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>Target prioritization</td>
<td>2</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Other on-topic</td>
<td>27</td>
<td>7%</td>
</tr>
<tr>
<td></td>
<td>Off-topic</td>
<td>4</td>
<td>1%</td>
</tr>
<tr>
<td>Answered correctly</td>
<td>Not at all</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Partially</td>
<td>11</td>
<td>3%</td>
</tr>
<tr>
<td></td>
<td>Completely</td>
<td>391</td>
<td>94%</td>
</tr>
<tr>
<td></td>
<td>With elaboration</td>
<td>1</td>
<td>0%</td>
</tr>
</tbody>
</table>
5.3.3.2 Experimental Outcomes

As Figure 12 shows, CFF training as conducted during the human study succeeded in making trainee CFFs faster and more accurate over time. Training decreased the time trainees needed to generate a CFF ($p < 0.01$), the number of CFFs that missed their target ($p < 0.01$), and the number of errors in CFF details ($p < 0.05$; marginally significant after Tukey HSD correction). Trainees’ target prioritization did not improve significantly. However, since training was effective according to several measures, the knowledge gaps and cognitive states observed during the study are likely to comprise the most important or most frequently occurring components of a useful ITS user model in this domain.

![Graphs showing experimental outcomes](image)

Figure 12: Trainee performance improved on several measures during the human study.

The average number of investigator observations per trainee was 33.25 ($\sigma = 10.2$). There were also 42 cognitive state self-reports collected during each training session. Together these represented a quantitative and qualitative improvement over the learner data usually available from simulator performance metrics in this domain (inputs per
trainee $\mu = 25.85$, $\sigma = 2.32$), which were additionally available for analysis. The additional observation data helped explore the knowledge states and cognitive states trainees are likely to experience during training in the CFF domain.

Of the 31 misconceptions or other knowledge gaps trainees were expected to experience, 25 were actually observed during the human study. In addition two new domain knowledge gaps, four new gaps relating to simulator usage, and three other gaps were observed at least once (Table 14). Overall, the participants’ experiences appeared to fit within the approximate number and detail level of the modeled gaps. The new gaps were used to improve the learner model, interventions, and initial instruction in the second human study (Chapter 6).

Table 14: New knowledge gaps that participants demonstrated.

<table>
<thead>
<tr>
<th>Knowledge Type</th>
<th>New Knowledge Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Does not know when the scenario is complete.</td>
</tr>
<tr>
<td></td>
<td>Believes moving targets must be timed.</td>
</tr>
<tr>
<td>Simulator</td>
<td>Does not know when the simulator is ready to accept another CFF.</td>
</tr>
<tr>
<td></td>
<td>Uses keyboard only with rangefinder, making it slower and less accurate.</td>
</tr>
<tr>
<td></td>
<td>Does not know when a target has been disabled.</td>
</tr>
<tr>
<td></td>
<td>Does not know the meaning or use of target labels.</td>
</tr>
<tr>
<td>Additional</td>
<td>Scenario context such as reasons for battle or for enemy behavior.</td>
</tr>
<tr>
<td></td>
<td>FDC terminology such as “Kilo,” “Shot,” or “Rounds complete.”</td>
</tr>
<tr>
<td></td>
<td>Experiment context such as what the experiment is trying to learn.</td>
</tr>
</tbody>
</table>

Figure 13 describes the distribution of trainees’ cognitive state reports. The most commonly reported cognitive states were flow (94% of reports in the mid- or high-range) and delight (79%). The reported occurrence of delight was both higher and more varied than hypothesized, suggesting its usefulness for a learner model if its impact on training effectiveness can be quantified. A reason for the difference might be that participants were actually reporting satisfaction, a state functionally similar to delight but with lower
magnitude (Graesser et al., 2006). The other hypothesized cognitive states also appeared often enough to allow modeling them: confusion (45%), surprise (42%), boredom (39%), and frustration (37%).

Self-reports of cognitive load showed little variability. Participants reported median values of cognitive load in most cases (63%). Instances of reporting cognitive load as high (24%) or low (13%) did not appear to correlate with any task or with trainee performance. The overall low variability of the cognitive load measure during this study suggests that it is less useful to model during CFF training. However, the usefulness of modeling cognitive load is likely to increase at more detailed timescales than this study recorded, such as second-to-second changes.

![Figure 13: Affective states trainees experienced during simulations varied, suggesting usefulness for a user model.](image)

Finally, participants answered an exit survey about their perceptions of the training experience. A positive result was that only a few participants in either condition reported the investigation distracted them from practice, suggesting that question-asking
can play a part in training without causing distraction. On the other hand, the distribution of responses was also the same between conditions for agreement that participants wanted to ask more questions during practice. This result is disappointing because encouragement to ask questions in one condition did not appear to change participants’ responses. Finally, one promising outcome was a different trend between conditions in answering whether the investigative method helped during training. Participants in the question-asking condition appeared more likely to agree that asking questions helped them as compared to participants in the think-aloud condition stating that thinking aloud during practice helped them. As before, trends were not subjected to statistical comparisons because of the small sample size.

5.3.4 Observational Study Summary

The human study of CFF training supported incorporating the hypothesized knowledge states and cognitive states into an ITS user model for the domain and suggested new knowledge gaps to incorporate as model states. Additional data collection would yield information about more model parameters such as state transitions, although for the purpose of this dissertation work those parameters will be set by subject-matter experts instead. The study also contributed general information about the characteristics of questions learners asked.
CHAPTER 6: THE INQUIRY MODELING POMDP ADAPTIVE TRAINER

The Inquiry Modeling POMDP Adaptive Trainer (IMP) is a vehicle for demonstrating and evaluating inquiry modeling and POMDP efficacy in controlling an adaptive training program. IMP is designed specifically to provide training support during Call for Fire (CFF) practice in the Deployable Virtual Training Environment (DVTE), described in Chapter 4.

The present chapter outlines IMP’s instantiation and domain-specific details. Building a program for specific, real-world training scenarios helps highlight the development process required to implement inquiry modeling and use a POMDP as an ITS learner model. Section 6.1 describes how trainees interact with IMP. Section 6.2 contains a high-level outline of the underlying IMP architecture, and Section 6.3 describes specifically IMP’s learner model. Section 6.4 gives initial information about the IMP prototype’s efficiency of development. Then, training human study participants and testing their performance (Chapter 7) evaluates IMP’s effectiveness.

6.1 Functional Description

6.1.1 Instruction and Practice

Before training begins, trainees receive pre-training education. IMP’s role is to provide support during practice, not to teach material for the first time. Trainees are expected to have the needed pre-training instruction before they use IMP and the practice
Various forms of educational media can fulfill this purpose, such as reading a text, watching a movie, or listening to a lecture.

IMP differs from some existing intelligent tutors in that IMP is not required to present every trainee with a whole curriculum, or indeed any particular required material. Instead, IMP selects only the material that will help a trainee during practice. Some well-known ITSs that also support learners only during practice rather than introducing new instructional material are PAT (Koedinger et al., 1997) and Andes (VanLehn et al., 2005).

After instruction, trainees practice the material in the FOPCSIM forward observer simulation. Descriptions of FOPCSIM and the CFF task appear in Chapter 4. Section 6.3 describes how the trainee’s interactions with the simulator change IMP’s learner model.

### 6.1.2 Interventions

During practice in the simulator, trainees receive two types of interventions: IMP-initiated hints and interaction with the QUI, an interface that lets trainees initiate help requests (Figure 14).
When IMP initiates a hint intervention, it appears as a floating window overlaid on the center of the simulator screen. Although the window is not modal and the trainee can move or resize it, in general it stays visible and obscures the simulator until the trainee uses the mouse to close it. Although this interface is quite distracting to the trainee’s workflow, it was designed in response to findings from the first human study that trainees frequently missed or ignored messages when they appeared in the top right corner of the screen and did not interrupt simulator use.

IMP is allowed to initiate an intervention only when certain events take place in the simulator: at the start of a practice session, after a trainee call for fire, after a trainee shot misses a target, and when a long period of time has passed without any call for fire.
These events signal moments when an intervention is likely to help the trainee, and, importantly, they give IMP the evidence it needs to update its learner model and determine which intervention will be effective (Section 6.3).

A hint window shows a Microsoft rich text format document, allowing for display of formatted text or a variety of embedded media. In IMP, all hints contain either text or a combination of text and graphics. Figure 15 shows an example of a hint intervention’s appearance.

Figure 15: An example of an IMP-initiated intervention, displaying the elaborated version of the RF hint.
6.1.3 Help Request Interface

A second channel for interactions between IMP and a trainee is IMP’s question user interface (QUI, pronounced kyew-ee). The QUI lets IMP perform inquiry modeling, that is, it gives learners a way to ask for specific help during training and uses any help requests to update its learner model. The QUI is always available during practice and cannot be moved or hidden.

IMP’s QUI is arranged in menu format. Since all questions must be chosen from a menu, help request freedom in IMP is lower than free-text entry but higher than the common hint button. Figure 16 illustrates the QUI part of the trainee interface. Up to six questions are listed vertically and a trainee may click any one to ask a question. Rather than requiring the user to scroll through a long list, questions are divided into five categories indicated by clickable tabs. When a trainee asks a question, IMP’s answer appears in a window with the same style as an IMP-initiated hint, above, but without the explanatory “Reminder” title.

Using a menu of pre-generated questions, as opposed to free question entry, scaffolds question-asking skills. The menu can include questions that would improve training but that a trainee might not think to ask otherwise, such as “What can I do to make my practice more effective?” Displaying questions for the trainee to consider eases help request formulation, which is known to be a difficult skill (e.g., Aleven & Koedinger, 2000; Walonoski & Heffernan, 2006). However, a menu interface does not necessarily address other parts of the help request process such as recognizing a need for help.
In addition to question formulation, the QUI question menu can scaffold question timing and recognition of when asking a question might be helpful. To accomplish this, IMP has the ability to highlight questions in the menu that it believes would probably provide useful help if the trainee asked them. Highlighted questions are displayed in a bolder font and have a light blue icon next to them. Menu groups that contain a highlighted question are also displayed with a similar highlight. Finally, the highlighted question also remains visible in the top box of the QUI, above the question menu, and a half-second, eye-catching animation accompanies its appearance there in order to draw visual attention. IMP decides when to highlight questions in the QUI using the same learner model and action recommendation process that control the hint interventions.

Figure 16: Using a menu interface for the QUI both lets trainees choose which questions they want to ask IMP and also implicitly suggests questions that trainees might not have considered otherwise.

6.1.4 Help Request Interface Characteristics

IMP’s QUI offers learners the freedom (Section 2.2.1) to select from a menu of questions, not just one or a few hint buttons as is common in current intelligent tutors. This mode of interface gives an intermediate level of freedom, although not complete freedom such as with full speech or text entry. With its increased freedom, the menu
interface increases the amount of information IMP’s learner model can draw from each help request.

Help requests through the QUI also employ moderate integration with IMP’s learner model (Section 2.2.2). While it is common in existing intelligent tutors to infer information from the fact of a help request and sometimes its timing, in IMP each help request can update the model based on both the question’s timing and its content. Each QUI item is associated in the model with evidence relating its content to an assessment of knowledge and cognitive states. IMP also considers question context since its POMDP learner model takes all previous interactions into account when interpreting new inputs. However, IMP does not integrate its assessments with information about question context in terms of concurrent simulator events or user interactions when the question is asked. The simulator does not report sufficient detail to IMP to let it integrate such context.

IMP’s responses to learner help requests (Section 2.2.3) place it in an authority relationship with the learner. IMP’s answers are not of the reference social role because some question answers are adaptive or trigger a rudimentary dialogue. For example, when a trainee asks, “Is that a friend or an enemy?” IMP will ask which target they mean. When a trainee asks, “How am I doing?” IMP will report on progress in the training session such as improved speed or accuracy. About a third ofQUI answers (9 out of 28) are calculated when the learner asks a question, rather than static. Finally, in its answers IMP does not manage an ongoing dialogue. Learner questions are considered separately and are not answered differently based on dialogue context.
Characterizing IMP’s functionality in relation to existing ITSs suggests that while IMP does not reach the maximum possible measure of each dimension, its user help requests are more free than most current systems and also better integrated than most. IMP combines greater freedom and model integration rather than offering one in isolation. IMP’s help request functionality was designed in all three dimensions to make inquiry modeling useful and to study its effects during training.

6.2 Component Structure

During simulator training and testing, IMP is always running but not always visible to the trainee. While IMP is running it observes information about trainee actions from FOPCSIM. Starting a practice scenario in FOPCSIM brings the IMP trainee interface on screen, and closing FOPCSIM hides the trainee interface. When IMP is not on-screen, an experimenter can control IMP with a Windows status area icon and context menu, shown in Figure 17. Experimenters can control other configuration information that changes infrequently by editing an XML configuration file.
Figure 17: IMP’s experimenter interface allows basic configuration changes.

Figure 18 shows a high-level view of IMP’s internal components and how they interact. The three most important components during training are modules for assessment, learner modeling, and pedagogy (strictly, andragogy). Supporting components that are required during model development are also shown.

Because the new state queue and observation chain representations do not require custom code to work, IMP’s POMDP controller is implemented with a generic, publicly available POMDP library. The Approximate POMDP Planning Library (APPL) implements the SARSOP policy search algorithm, as well as functions for loading and maintaining a POMDP and belief (Wei, 2010). Calls to APPL functions are stateless, blocking, and in general must return quickly. They appear in Figure 18 as double-headed arrows. APPL is written in native C++, and IMP uses managed C++ wrappers for
interoperation with APPL. IMP is written in C#, with interface components built in Windows Presentation Foundation.

![Diagram of IMP component interactions](image)

**Figure 18:** IMP component interactions. Rectangles represent components, with shaded rectangles indicating components that are specific to the CFF domain and white rectangles indicating generic, reusable components. Arrows indicate control flow during training. Double-headed arrows indicate calls to the APPL generic POMDP library.

Control in IMP is event-driven. When new evidence about a trainee is available, IMP first processes the evidence in an assessment module, next consults its learner model, and finally interprets the learner model’s recommendation in a pedagogical module. This section describes the data flow in more detail.

First, there are three main sources of evidence or observations in the CFF domain (Section 6.3.4). Information can come from IMP’s trainee interface, such as noting when
a trainee read a hint. The simulator generates a variety of observations about trainee performance. The passage of time can also act as evidence, when trainees pause for a long time or ask several questions quickly. The role of the assessment module is to translate these observations into evidence that the learner model can use. For example, the length of a pause is a continuous variable that this module assesses as either acceptable or too long. When the simulator reports the coordinates where a trainee’s shot fell, the module assesses many aspects of the shot’s target selection and engagement. For example it determines whether the second target prioritization rule was correct, slightly incorrect, quite incorrect, or indeterminable because of a confounding trainee error.

Next, once raw information is transformed into assessments, IMP consults its learner model. IMP interprets each assessment in the context of its current belief and the last action it took. Standard Bayesian inference allows it to bring its belief up to date in light of the new evidence. The updated belief is sufficient to reconstruct all previous actions and observations, so IMP does not need to retain any explicit history. IMP then consults its policy and finds the optimal next action given its refined belief. The policy is constructed to include planning ahead and the control cycle reconfirms the plan at each step, so IMP also does not need to store or commit to a plan. The components pertaining to IMP’s POMDP learner model are generic and the same approach is reusable for other POMDP ITSs. Section 6.3.2 discusses the domain-specific states that IMP models.

Finally, the recommendation from IMP’s learner model is filtered and implemented in a pedagogical module (Section 6.3.3). The pedagogical module makes domain-specific decisions about how to carry out the recommendation with the
interventions available to IMP by selecting specific hint text to display or changing the QUI to highlight questions. The pedagogical module can also contain absolute rules that override certain recommendations for pedagogical reasons. In IMP, only one such rule exists—the pedagogical module prevents IMP from displaying any material before it is first introduced in the pre-practice instructional material.

Several external modules were also created to support the development of IMP and similar POMDP ITSs. An answer editor is specific to the current domain and helps with authoring question answers and hint popups. A POMDP creator translates model parameters that can define any domain into a fully defined POMDP in a standard format (Section 6.4.1), so that a publicly available offline policy search program in the APPL library can output an optimal policy for that POMDP (Section 6.4.2). Finally, a program for applying a policy to specific inputs, either chosen singly by a user or generated many thousands of times according to a configurable model, provides a useful testbed and platform for producing simulated results.

### 6.3 POMDP Learner Model

This section details the contents of IMP’s learner model. It qualitatively describes information developed during the first human study to help understand the trainees’ likely mental states during practice, what actions IMP can take to support effective practice, and what information IMP can use to accomplish its goals. Next, Section 6.4 discusses the process of developing the learner model.
6.3.1 Simulator Limitations

IMP’s learner model is designed to support trainees in practicing specific tasks, and changing the target tasks would require changing the learner model. Training for the CFF task can be divided into two topics: simulator usage and domain tasks. Domain tasks were described in Chapter 4. However, before practicing any domain material, trainees must learn the steps to call for fire in the simulator (so-called “button-ology”). In particular, the many controls in the radio menu often confuse first-time users.

Importantly, a trainee’s simulator usage is not reported to IMP on a finely detailed level. Instead, the simulator only reports a few events, such as a completed call for fire. The simulator does not report information about trainee mistakes, and if a trainee fails to complete a call for fire, the simulator does not report at all. As examples, IMP has no information about when trainees set individual radio controls, and likewise IMP cannot react immediately when a trainee misuses the rangefinder tool and reads off a wrong location. The simulator will not give IMP any information about any of these low-level actions until much later in the training session, if at all.

Because the simulator reports useful evidence only when the trainee has mastered the simulator usage skills enough to complete a call for fire, IMP’s only information about why a trainee is having trouble before that stage comes from QUI interaction. The QUI does help trainees with simulator usage, but when trainees ask for help with the QUI, their knowledge gaps are cleared through those interactions directly. When trainees have problems with simulator usage but do not ask for help, outside of the few modeled gaps, there is no evidence IMP can use to guess what the problem is and therefore no
value in modeling all the possible problems. Therefore, IMP models only a few ways trainees can make usage mistakes. Instead, IMP’s learner model focuses on trainee tasks within the CFF domain.

6.3.2 Model States

Based on an analysis of the training tasks and the KSAs required to complete them, 17 knowledge gaps were identified that IMP should model (Table 15). Of these, seven relate to target selection, and seven relate to target engagement. Only three states relate to simulator usage. IMP represents knowledge gaps with a state queue for efficiency. In the queue, target selection gaps are mostly prioritized above target engagement gaps because target selection needs to be completed before target engagement begins and because certain selection errors preclude assessing any engagement errors. However, the engagement gap RF has a higher priority in the queue because errors related to rangefinder usage can make the simulator misinterpret which target a trainee is trying to engage.
Table 15: The 18 knowledge gaps in IMP’s learner model are listed in state queue order.

<table>
<thead>
<tr>
<th>Gap Name</th>
<th>Description</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>Radio use -- missed a &quot;k&quot; during CFF transmission</td>
<td>Usage</td>
</tr>
<tr>
<td>Dir</td>
<td>Radio use -- invalid azimuth, e.g. not four digits</td>
<td>Usage</td>
</tr>
<tr>
<td>Dis</td>
<td>Radio use -- invalid range, e.g. not a number</td>
<td>Usage</td>
</tr>
<tr>
<td>RF</td>
<td>Rangefinder use</td>
<td>Engage</td>
</tr>
<tr>
<td>Friend1</td>
<td>Does not know target type 1 is a friend</td>
<td>Select</td>
</tr>
<tr>
<td>Friend2</td>
<td>Does not know target type 2 is a friend</td>
<td>Select</td>
</tr>
<tr>
<td>Foe1</td>
<td>Does not know target type 3 is an enemy</td>
<td>Select</td>
</tr>
<tr>
<td>Foe2</td>
<td>Does not know target type 4 is an enemy</td>
<td>Select</td>
</tr>
<tr>
<td>Priority1</td>
<td>Does not prioritize moving targets</td>
<td>Select</td>
</tr>
<tr>
<td>Priority2</td>
<td>Does not prioritize closer targets</td>
<td>Select</td>
</tr>
<tr>
<td>Fire1</td>
<td>Does not know fire pattern for a moving target</td>
<td>Engage</td>
</tr>
<tr>
<td>Fire2</td>
<td>Does not know fire pattern for a static target</td>
<td>Engage</td>
</tr>
<tr>
<td>Desc1</td>
<td>Does not know description for enemy 1</td>
<td>Engage</td>
</tr>
<tr>
<td>Desc2</td>
<td>Does not know description for enemy 2</td>
<td>Engage</td>
</tr>
<tr>
<td>Ammo1</td>
<td>Does not know munition type for enemy 1</td>
<td>Engage</td>
</tr>
<tr>
<td>Ammo2</td>
<td>Does not know munition type for enemy 2</td>
<td>Engage</td>
</tr>
<tr>
<td>Hurry</td>
<td>Thinks scenario is complete, but enemies still remain</td>
<td>Select</td>
</tr>
<tr>
<td>None</td>
<td>A special state that marks the end of the queue, when no more knowledge gaps exist</td>
<td>N/A</td>
</tr>
</tbody>
</table>

IMP models the same four affective states as used elsewhere in the present dissertation: boredom (BOR), confusion (CON), flow (FLO), and frustration (FRU).

Cognitive states are orthogonal to knowledge states and IMP’s POMDP learner model represents them in a fully enumerated form. Transitions between cognitive states depend on whether IMP’s last intervention was helpful to the trainee, unhelpful, or a no-op or ignored. When an action is helpful, that is, it clears an actual knowledge gap, the trainee is more likely to transition into the flow state. When an action is unhelpful, the trainee is more likely to become bored or frustrated. Actions highlighting QUI items, though, have no unhelpful transitions—they can only be helpful or ignored.
6.3.3 Recommendations

In IMP, interventions can be in the form of either a hint display or a QUI highlight. The learner model recommends interventions and the pedagogical module carries them out.

The pedagogical module has available to it two versions of most messages: elaborated and brief. When IMP decides to present the same hint more than once, it cycles through all available versions of the message, showing a different version each time in order to increase the chances a repeated action will be effective and reduce learner frustration with the repetition. An elaborated version of a hint might be four sentences giving a detailed explanation of when, how, and why to correct a behavior, while a brief version of the same hint might simply remind the learner of what to do in a few impactful words. The hint system differs from other ITSs that offer multiple support levels, progressing from general hints to a bottom-out hint. The reason is that there were not planned to be enough hinting opportunities in one training session to progress through multiple support levels.

An example of different hint versions might be IMP tutoring recognition of combatant vehicles with a simplified description of a target’s hull, armament, turret, suspension (HATS) characteristics. First IMP would address a particular target description mistake with the elaborated hint “Use the target description BMP. [Picture of a BMP] Look for the boat-like front and flat top with a small turret.” If IMP needed to present the same hint again later, the pedagogical module would select the brief version
“Target description is BMP [Same picture].” Table 16 lists an example of hint text for addressing each modeled gap.

Table 16: Example text of hints that address each gap. IMP has at least a brief hint and an elaborated hint to address each modeled gap.

<table>
<thead>
<tr>
<th>Gap</th>
<th>Brief Hint Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>Did you press the &quot;K&quot; button on every line?</td>
</tr>
<tr>
<td>Dir</td>
<td>DIR should be four digits.</td>
</tr>
<tr>
<td>Dis</td>
<td>Put the distance in DIS.</td>
</tr>
<tr>
<td>RF</td>
<td>Put the center of the rangefinder on your target and right-click.</td>
</tr>
<tr>
<td>Friend1</td>
<td>Friend -- do not attack (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Friend2</td>
<td>Friend -- do not attack (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Foe1</td>
<td>Enemy -- disable all of these (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Foe2</td>
<td>Enemy -- disable all of these (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Priority1</td>
<td>Attack all moving targets first.</td>
</tr>
<tr>
<td>Priority2</td>
<td>Attack the closest enemy first.</td>
</tr>
<tr>
<td>Fire1</td>
<td>Attack moving targets with FIRE FOR EFFECT.</td>
</tr>
<tr>
<td>Fire2</td>
<td>Attack stationary targets with ADJUST FIRE.</td>
</tr>
<tr>
<td>Desc1</td>
<td>Target description is BMP (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Desc2</td>
<td>Target description is BTR-70 (image highlighting HATS markers)</td>
</tr>
<tr>
<td>Ammo1</td>
<td>Attack the BMP with ICM.</td>
</tr>
<tr>
<td>Ammo2</td>
<td>Attack the BTR-70 with HE/QUICK.</td>
</tr>
<tr>
<td>Hurry</td>
<td>Attack enemies quickly until they are all disabled.</td>
</tr>
</tbody>
</table>

In addition to direct hints, IMP can also recommend highlighting a QUI question as a less intrusive way of addressing each of the knowledge gaps except RF. Since negative affective states are believed to hinder learning, IMP has one hint recommendation and two highlight actions that address negative affect and attempt to help the learner feel more positive during practice. Finally, IMP has a no-op recommendation that it can use to skip a turn if its model is uncertain about whether the other actions would be helpful or harmful.

In designing the range of actions IMP can recommend, it is less useful to create hint granularity different than that of the knowledge gap model, such as creating several...
actions that tutor different fine-grained issues but all relate to a single gap or using a single action to tutor several gaps. Especially if the former mismatch arises, it is a signal that the model should contain gaps with sufficient detail to support the desired interventions. However, in the future it might be useful to create multiple recommendations that address a single gap if a cognitive state also helps determine whether one hint or another would help a learner more.

As a final note about IMP’s actions, the pedagogical module does contain the ability to change, not just interpret, the learner model’s recommendations. Specifically, IMP has one pedagogical rule guarding against presenting during-practice hints about material before the material has been appeared in introductory exposition. If the learner model makes such a recommendation, the pedagogical module changes it to a no-op action. No other rules were implemented outside the learner model in order to give the POMDP maximum latitude to perform in the present evaluations. In the future, it would be possible to implement other rules that domain experts find useful such as rules against repeating hints or skipping hints.

An additional recommendation type that might be useful in the future could be an IMP-initiated question type. Such an action could help IMP determine whether a gap exists or not, triggered by the POMDP in response to model ambiguity. Naturally such questions could be immediately followed by the correct answer, thus tutoring the gap if it did exist. In this way IMP-initiated questions could be modeled in the same structure as existing hints.
6.3.4 Evidence

Evidence for updating IMP’s learner model comes from trainee performance in the FOPCSIM simulator and interactions with IMP’s QUI. IMP’s assessment module translates these data into evidence for or against the presence of particular knowledge and cognitive states in the learner model. Table 17 lists the types of information IMP collects and which knowledge gaps they relate to, while Table 18 shows how the data relate to the modeled cognitive states.

Table 17: Observations IMP can use to infer a trainee’s current knowledge state. Observation chains let IMP represent observations containing several dimensions of additional information or multiple values. Each observation gives evidence for or against the presence of the related knowledge gaps.

<table>
<thead>
<tr>
<th>Observation</th>
<th>Additional Information</th>
<th>Values</th>
<th>Related Gaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long Idle</td>
<td>None</td>
<td></td>
<td>K, Hurry</td>
</tr>
<tr>
<td>CFF</td>
<td>None</td>
<td>F / T</td>
<td>K, Hurry</td>
</tr>
<tr>
<td>Bad DIR</td>
<td></td>
<td>F / T</td>
<td>Dir, RF</td>
</tr>
<tr>
<td>Bad DIST</td>
<td></td>
<td>F / T</td>
<td>Dist, RF</td>
</tr>
<tr>
<td>Target Type</td>
<td></td>
<td>1 / 2 / 3 / 4</td>
<td>Friend1, Friend2, Foe1, Foe2</td>
</tr>
<tr>
<td>Priority1</td>
<td>F / T / TT / U</td>
<td></td>
<td>Foe1, Foe2, Priority1</td>
</tr>
<tr>
<td>Priority2</td>
<td>F / T / TT</td>
<td></td>
<td>Foe1, Foe2, Priority2</td>
</tr>
<tr>
<td>Fire1</td>
<td>F / T / U</td>
<td></td>
<td>Fire1</td>
</tr>
<tr>
<td>Fire2</td>
<td>F / T / U</td>
<td></td>
<td>Fire2</td>
</tr>
<tr>
<td>Desc1</td>
<td>F / T / U</td>
<td></td>
<td>Desc1</td>
</tr>
<tr>
<td>Desc2</td>
<td>F / T / U</td>
<td></td>
<td>Desc2</td>
</tr>
<tr>
<td>Ammo1</td>
<td>F / T / U</td>
<td></td>
<td>Ammo1</td>
</tr>
<tr>
<td>Ammo2</td>
<td>F / T / U</td>
<td></td>
<td>Ammo2</td>
</tr>
<tr>
<td>Shot Miss</td>
<td>None</td>
<td></td>
<td>K, Dir, Dis, RF, Hurry</td>
</tr>
<tr>
<td>Help Request Read</td>
<td><strong>Topic</strong></td>
<td>Usage / Steps / IFF / Priority / Fire / Desc / Ammo / Affect / Extra</td>
<td>Mixed or all gaps</td>
</tr>
<tr>
<td>Help Request Skipped</td>
<td><strong>None</strong></td>
<td></td>
<td>No gaps</td>
</tr>
</tbody>
</table>

Table 17: Observations IMP can use to infer a trainee’s current knowledge state. Observation chains let IMP represent observations containing several dimensions of additional information or multiple values. Each observation gives evidence for or against the presence of the related knowledge gaps.
Table 18: Observations IMP can use to infer a trainee’s current cognitive state. Plus (+) means a state is more likely and minus (−) means a state is less likely after the observation, while 0 indicates no change except for relative changes.

<table>
<thead>
<tr>
<th>Observation</th>
<th>BOR</th>
<th>CON</th>
<th>FLO</th>
<th>FRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long Idle</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>CFF</td>
<td>−</td>
<td>0</td>
<td>0</td>
<td>−</td>
</tr>
<tr>
<td>Shot Miss</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Help Request</td>
<td>0</td>
<td>0</td>
<td>−</td>
<td>0</td>
</tr>
<tr>
<td>Help Request Read</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
</tr>
</tbody>
</table>

The practice simulator transmits information after each trainee call for fire and when each trainee shot lands. In particular, a completed CFF can contain information about many knowledge states. Observation chains allow transmitting this evidence to the learner model. However, in the absence of a CFF, IMP must rely on its QUI for information about trainees’ condition and needs.

Table 19 lists the text of questions placed in the QUI menus for learners to ask. IMP interprets help requests as evidence against the presence of the related knowledge gaps if the learner reads IMP’s answer. If the learner does not read the answer, IMP interprets the help request as a mistaken click or looking for other help than was provided, evidence of boredom or frustration, and does not make any inference about knowledge changes. Whether a learner has read an answer is determined by whether the answer was open for at least two seconds plus 0.02 seconds per character in the answer’s text. This figure is an estimate including time needed to read and understand text along with the average time to use the mouse to close the window.

The process of choosing content for the QUI depended on considering frequently asked questions during the first human study. The questions were filtered so as to support
the scaffolding opportunities of the menu interface. For example, some common
questions were off-topic and were therefore not included in the QUI as an attempt to
courage trainees to focus on adaptive help requests. While it is certainly possible for
experts to select the questions that should appear in the QUI, care should be taken make
sure they are aware of the difference between questions experts think learners would ask
and questions novices really do ask (Chi et al., 1981).

Table 19: Questions trainees can ask IMP. Menu refers to the display group tab where the question appears in the
QUI, and during simulator usage training IMP only displays the Radio Menu and Training menu tabs. Topic refers to
which set of knowledge gaps IMP makes inferences about when a trainee asks a question.

<table>
<thead>
<tr>
<th>Menu</th>
<th>Question</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Info</td>
<td>Is that a friend or an enemy?</td>
<td>Select – IFF</td>
</tr>
<tr>
<td></td>
<td>Is that the highest priority?</td>
<td>Select – Priority</td>
</tr>
<tr>
<td></td>
<td>What is the warning order?</td>
<td>Engage – Fire</td>
</tr>
<tr>
<td></td>
<td>What is the target description?</td>
<td>Engage – Desc</td>
</tr>
<tr>
<td></td>
<td>What is the method of engagement?</td>
<td>Engage – Ammo</td>
</tr>
<tr>
<td></td>
<td>Did I destroy it?</td>
<td>Usage</td>
</tr>
<tr>
<td>Radio Menu</td>
<td>Why do I have to wait so long?</td>
<td>Usage</td>
</tr>
<tr>
<td></td>
<td>What do I put in the menu boxes?</td>
<td>Usage</td>
</tr>
<tr>
<td></td>
<td>What do I do after End of Mission?</td>
<td>Select – Steps</td>
</tr>
<tr>
<td></td>
<td>Why can’t I go back?</td>
<td>Extra</td>
</tr>
<tr>
<td>How Do I…?</td>
<td>How do I know which targets are enemies?</td>
<td>Select – IFF</td>
</tr>
<tr>
<td></td>
<td>How do I pick the enemy to attack first?</td>
<td>Select – Priority</td>
</tr>
<tr>
<td></td>
<td>How do I hit a moving target?</td>
<td>Engage – Fire</td>
</tr>
<tr>
<td></td>
<td>How do I know the Target Description?</td>
<td>Engage – Desc</td>
</tr>
<tr>
<td></td>
<td>How do I pick the Method of Engagement?</td>
<td>Engage – Ammo</td>
</tr>
<tr>
<td></td>
<td>How do I know when the scenario is done?</td>
<td>Select – Steps</td>
</tr>
<tr>
<td>Training</td>
<td>How can I avoid making mistakes?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>How can I get the best score?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>How am I doing?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>How much have I improved?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>How can I learn better?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>How can I get done faster?</td>
<td>Extra</td>
</tr>
<tr>
<td>What If…?</td>
<td>What if enemies attack my position?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>What if we have to protect another unit?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>What if we get help from air support units?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>What if more enemies appear?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>What if a target behaves differently?</td>
<td>Extra</td>
</tr>
<tr>
<td></td>
<td>What if we need to attack a new type of target?</td>
<td>Extra</td>
</tr>
</tbody>
</table>
6.4 Learner Model Design

Although IMP was a prototype and therefore required greater effort, discussing its development process may suggest how efficient to build future ITSs could be.

6.4.1 POMDP Parameters

IMP’s state queue and observation chain representations somewhat determine its model structure. Their compactness and tractability arise from certain independence assumptions which should be approximately true in order to apply the representations (Chapter 3). Since the CFF practice domain meets the basic assumptions, building its learner model largely consists of assigning appropriate parameter values. The model parameters are simpler than probability tables in an unconstrained POMDP, and they can be framed as four descriptions that potentially can be completed by subject-matter experts (SMEs) rather than computer experts or engineers. The four model descriptions are two-dimensional, real-valued matrices describing gap initial probability, gap dependence, base action efficacy, and observation multipliers.

To define gap initial probability, an expert must decide for each knowledge gap the percentage of trainees for which that gap will be present at the start of training. The assumption that gaps are not correlated simplifies this process so that only one percentage needs to be generated for each knowledge gap. In IMP, gap initial probability was estimated from gap occurrence during the first human study. The gap initial probability contributes to POMDP state transitions and the initial belief.
Gap dependence is the value $d$ in Section 3.1. For each pair of knowledge gaps, an expert must decide how much less likely the lower-priority gap is to be cleared when the higher-priority gap is present. The study with simulated learners indicated performance is less sensitive to this parameter, and in IMP the values were estimated by a SME. Base action efficacy is the value $a$ in Section 3.1, and it describes the marginal likelihood, independent of cognitive and knowledge state, that each hint will clear each gap if it is present. This value should be estimated by the hint author, presumably in conjunction with a SME. The gap dependence and action efficacy both contribute to POMDP state transitions.

Finally, observation multipliers describe how the presence of each gap or cognitive state makes each observation more or less likely. These are always positive values, with fractions indicating an observation is less likely and values greater than one when a gap’s presence tends to make an observation more likely. Framing the description in this way is intended to make observations easier for non-mathematicians to estimate. Internally, the observation multipliers are converted into a standard POMDP observation emission table.

In IMP, the four descriptions are based on expert estimates and the findings of the first human study. The amount of time invested in eliciting the parameters was high, since it included both expert consultation and all the effort required to design and execute a study with human participants. In the end, most of IMP’s model descriptions were drawn from expert input, showing that a human study is not strictly required to create useful descriptions.
Finally, the effort of translating IMP’s descriptions into a model was small, because a POMDP creation program can read in description tables and output standard POMDP files. It can calculate models the size of IMP’s exactly, and can approximate large models the size of those in Section 5.2 with Monte Carlo methods. The POMDP creation program is a product of the present dissertation. Making model creation easier still is an ongoing goal which, in the future, might include giving parameter elicitation a graphical interface useful to SMEs, or supplementing expert input with machine learning from actual learner data.

6.4.2 Policy Search

After the creation of IMP’s POMDP, the final step to prepare IMP for use is finding a policy that solves the modeled problem. Since the new representations do not require a specialized policy search algorithm, the APPL implementation of the SARSOP algorithm was used (Wei, 2010). In IMP’s case, a policy search was run for approximately 48 hours on a single core of a 3-GHz processor with 8 GB of RAM before reaching an arbitrary time limit. The search was not multithreaded. Policy search ran without converging for longer than is usual in some non-tutoring applications. A likely reason is that policy search with observation chains requires a long horizon, looking deep into the future to estimate the values of belief points. Even when a tutoring session only lasts for a few dozen turns at most, the POMDP might run for hundreds of turns because each observation chain is input over several turns. A generic policy search that does not
have knowledge of the domain must consider all these possible observations, including combinations that actually will never occur.

Figure 19 shows the progress over time of the search for IMP’s tutoring policy. As will be described in Section 7.1.2, IMP was used in two different experimental conditions, requiring two different models. Figure 19 shows both models because there are a few interesting differences between them. One version is the full IMP model, and a simplified version removes all inputs and actions relating to the QUI.

In Figure 19, the number of beliefs visited measures how thoroughly SARSOP prepared for tutoring. SARSOP is point-based and estimates the contours of the optimal policy by sampling points in belief space (Section 2.3.3). When IMP reaches points that SARSOP never anticipated during policy search, it must approximate using nearby points that were searched. By the end of 48 hours, the policy search on the ablated model with no QUI had estimated values for 1323 points in belief space, compared to 609 points in the policy search on the full model. Other measures of search coverage such as tree traversals and value backups were also approximately doubled without the QUI in the model. The greater size of the full POMDP slowed policy search. Furthermore, since search slowed over time on both models, SARSOP may never be able to explore the more complex belief spaces like that of the full IMP POMDP as thoroughly as it explores the simpler spaces like the ablated one.

A second measure of search progress in Figure 19 is the size of the policy over time. Alpha (α) vectors in a policy partition the belief space. A policy containing more undominated α vectors projects a more complex partitioning and, therefore, displays
more behavior differences depending on finely differentiated beliefs. While more $\alpha$ vectors are not necessarily useful in practical application because some may never be activated while controlling an ITS, more $\alpha$ vectors signal an increased upper bound on an ITS’s ability to adapt to a learner’s needs. For both the full and ablated IMP models, search produced policies composed of between 2,000 and 2,500 $\alpha$ vectors. The large policies gave IMP the potential to display responsive rather than simplistic behavior. The decrease in policy size at the end of one search indicates a simpler policy was found that produced better expected reward than the previous, more complex one. Both generated policies were small enough to execute in sub-second time during practice.

Finally, policy search may also be characterized by the bounds on the estimated value of the output policies. Figure 20 illustrates how these bounds changed over time during policy search. In both models, the difference between upper and lower bounds on expected reward narrowed quickly in the first seconds of policy search. It might be interesting in the future to check the actual outcomes of policies that were generated in just a few seconds. However, while the change in the bounds on expected reward quickly slowed, it should not be assumed that the policies were not improving. The small increases in expected reward during that time corresponded to complexification of behavior (Figure 19) that was likely to be important in effectively adapting to learners and producing positive outcomes for real participants.
Figure 19: Policy search on the two IMP models progressed over the entire 48 hours allowed it. Both policies grew to about 2,000 alpha vectors, letting IMP make about 2,000 different decisions based on learner interactions. Policy search visited fewer points in belief space with the full IMP model because its greater complexity slowed the search.
During policy search, reward estimates improved quickly for a few seconds and slowly thereafter. However, the small improvements might not mean useful search ended, since policy complexity was still increasing (Figure 19). The time axis is on a log scale to emphasize the first seconds.

Figure 20: During policy search, reward estimates improved quickly for a few seconds and slowly thereafter. However, the small improvements might not mean useful search ended, since policy complexity was still increasing (Figure 19). The time axis is on a log scale to emphasize the first seconds.
6.5 Conclusion

This chapter describes the Inquiry Modeling POMDP Adaptive Trainer (IMP). IMP supports trainees during practice in a simple but realistic call-for-fire task. IMP implements the knowledge representations introduced in the present dissertation to make the job of intelligent tutoring tractable for a POMDP. IMP also uses inquiry modeling to gain new information from trainees’ help requests that is not available from assessing their performance alone. Building and testing IMP provides insight into these ideas’ potential for providing efficient and effective instruction. The following chapter describes the positive and promising outcomes of evaluating IMP in a human study.
CHAPTER 7: SUMMATIVE EVALUATION

IMP, an adaptive trainer implementing inquiry modeling with a POMDP learner model, was evaluated in a human study. IMP trained participants to perform a call for fire (CFF) in a U.S. Marine Corps virtual training environment. The purpose of the evaluation was to learn about how inquiry modeling interacts with human learners and determine whether inquiry modeling could drive effective instruction on a realistic learning task.

7.1 Method

7.1.1 Materials and Procedures

Participants \( N = 126 \); c.f. Section 7.2.1) were United States citizens at least 18 years old. The training simulator is listed on the United States Munitions List (USML), and as such all study participants were required to be United States citizens for compliance with International Traffic in Arms Regulations (ITAR). In order to participate in the study with fluency, participants required unimpaired or corrected abilities to see, hear, read and write in English, and manipulate a keyboard and mouse for extended periods of time.

Participants were required to have no military training or experience. During previous studies in the CFF domain (Vogel-Walcutt et al., 2009; Vogel-Walcutt et al., 2008), individuals with military backgrounds performed disproportionately well compared to civilians. The difference appeared even when the military experience was not related to the CFF domain and was probably caused by general framing schemata that
helped the military personnel process any domain training more readily than civilians possessing no such frame of reference (Bartlett, 1932; Daley, 1999).

Experimental sessions lasted up to two hours and participants were compensated $20 at the end of the session. Each session had between one and eight participants. Each participant worked on one laptop from a set of eight laptops arranged around a large table in a closed room. Participants worked individually but in view of each other.

Trainee laptops were configured with version 3.5.2 of the Deployable Virtual Training Environment (DVTE) Combined Arms Network (CAN) including the FOPCSIM forward observer PC simulator. Laptops also had an external mouse and headphones. An IMP instance ran locally on each laptop. Trainee laptops were remotely controlled by an experimenter with an Instructor Support Station, a software suite that lets military trainers coordinate and monitor training (Schatz, Champney, Lackey, Oakes, & Dunne, 2011). In addition to laptops, participants were given pen and paper at the start of each session and instructed to take notes.

After collecting participants’ informed consent and demographic information, each experimental session proceeded according to the agenda in Table 20. First, participants learned new material and practiced in the simulator twice. Each practice scenario was preceded by a three-minute introductory video teaching the material the trainees needed to practice. The simulator usage video was also reinforced with a set of three ungraded practice worksheets in order to address many of the misconceptions observed during the first human study. After each simulator practice, participants completed surveys on their perceived cognitive load and affective states, the same
instruments as those described in Section 5.3. In the first scenario, trainees were instructed on and practiced using the simulator to issue a call for fire (Section 4.1). The scenario contained two stationary enemy targets and no friendly targets. Trainees were given ten minutes to practice. In the second scenario, trainees learned about and then practiced target selection and engagement (Section 4.2). The battlefield in this scenario contained four of each of the four target types: eight friendly targets and eight enemy targets. Half of the friendly and enemy targets were moving in repetitive patterns, and half were stationary. Trainees were given sixteen minutes to practice.

After practice, two ten-minute simulator scenarios tested participants’ performance in calling for fire. Finally, participants completed a written test on the practiced material and a survey with 24 questions about their perceptions of the simulator practice. These instruments are described in Section 7.1.3.

Table 20: Summative study agenda.

<table>
<thead>
<tr>
<th>Agenda Task</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2 minutes</td>
</tr>
<tr>
<td>Consent and background forms</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Simulator introduction</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Simulator usage worksheets</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Simulator practice (three experimental conditions)</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Domain task introduction</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Domain task practice (three experimental conditions)</td>
<td>16 minutes</td>
</tr>
<tr>
<td>Domain task tests</td>
<td>20 minutes</td>
</tr>
<tr>
<td>Written test</td>
<td>15 minutes</td>
</tr>
<tr>
<td>User perceptions survey</td>
<td>5 minutes</td>
</tr>
<tr>
<td>Debrief</td>
<td>3 minutes</td>
</tr>
</tbody>
</table>
7.1.2 Treatments

The IMP evaluation used a between-subjects design in order to avoid carryover effects. Participants were randomly assigned to one of three experimental conditions. The conditions received the same introductory material and simulator scenarios but differed in the kind of support participants received during practice. Participants in condition O, the control condition, trained with expository videos only and received no support during practice. Condition O represented the status quo for U.S. Marine Corps trainees’ use of FOPCSIM for practice. Participants in group Q underwent the same experimental procedure but received support from IMP during the 26 minutes of the two simulator practice sessions. Finally, participants in group P underwent the same experimental procedure but received practice support from an ablated version of IMP. The ablated IMP maintained a POMDP learner model and provided adaptive practice support, but did not display a QUI and had no inquiry modeling functionality.

Table 21: Experimental design for IMP’s evaluation.

<table>
<thead>
<tr>
<th>Condition O (Control)</th>
<th>Condition P</th>
<th>Condition Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect informed consent and background information.</td>
<td>Train without adaptive help—status quo.</td>
<td>Measure performance on simulator tasks and written test, along with user perceptions.</td>
</tr>
<tr>
<td></td>
<td>Train with the ablated IMP—no QUI and no inquiry modeling.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Train with the full IMP.</td>
<td></td>
</tr>
</tbody>
</table>

7.1.3 Instruments

The IMP evaluation employed several instruments. First, after each of the four simulator scenarios participants completed questionnaires about their cognitive load and
affective state while using the simulator; Section 5.3 describes these two questionnaires. Second, after simulator practice participants demonstrated their skills in two scored simulator scenarios. Finally, participants completed a written knowledge test and a survey about their training experience.

The IMP evaluation did not employ a pre-training test. In previous studies using the CFF domain (Vogel-Walcutt et al., 2009; Vogel-Walcutt et al., 2008), pre-training tests were found to affect training by acting as advance organizers. In the place of a pre-training test all participants were assumed to be starting with the same knowledge and to have no experience related to the tested material. In order to support this assumption, no one with any military background was allowed to participate in the study and intake forms asked participants about any previous experiences that might relate to the study.

After training, two simulator scenarios tested participants’ performance in calling for fire. The scenarios were limited to ten minutes each in order to add time pressure and guard against any ceiling effect. The first scenario presented the same battlefield situation as the practice, with targets in different locations. This scenario tested the target KSAs in the same setting in which they had practiced. The second scenario tested the trainees’ ability to apply the practiced KSAs in a new setting. Trainees were presented with another battlefield configured as before but were directed to act out the role of an opposing forward observer located on the other side of the battlefield. In order to carry out these directions and perform well on the second test, participants needed to infer behaviors they had not been explicitly taught. Participants needed to change their target selection by inverting their friend-or-foe recognition and by prioritizing enemies
according to their proximity to a location other than their own, while not changing their prioritization of moving targets before stationary ones. Participants also needed to extrapolate their target engagement rules to determine which munitions to request for the new target types, while not changing the practiced rules about firing patterns. This scenario was redesigned after the first human study to vary more aspects of the practiced material and emphasize training transfer to a different situation. Participants who generalized their target selection and engagement correctly to fit the second test scenario demonstrated a deeper understanding of the practiced material, rather than mere rote memorization.

In the two tests, simulator reports on target selection and target engagement were scored separately. For each scenario, a participant’s target selection score reflected the percentage of CFFs that correctly attacked the highest-priority enemy, accounting for identification of friend or foe and the two target prioritization rules. The score was not affected by whether the requested fire hit the intended target or not. The target engagement score reflected the percentage of CFFs attacking enemy targets that included the correct target description, firing pattern, and munition for the particular target. The simulator was set so that all shots destroyed nearby targets, even when the participant transmitted incorrect engagement information.

Finally, after performing in the simulator tests participants completed a 19-question written test of their declarative knowledge of the practiced material. This instrument was redesigned after the first human study to thoroughly test the practiced target selection and engagement KSAs. Ten of the test items targeted recall of material
from the introductory videos, and the other nine items required application of the material to new situations the videos did not address directly. Four of the nine extension items were in a short-answer format, and the other 15 test questions were multiple-choice. Participants also completed a survey with 24 user-acceptance questions about IMP. The user-acceptance items were designed to elicit responses in five clusters: self-efficacy, task value, training perception, trainer ability perception, and trainer outcome perception. These items were collected with a seven-point Likert scale. Item text referred to the whole experience of simulator practice, so that participants did not need to differentiate IMP facilities from FOPCSIM and participants in the control condition did not need different item text. The full text of the user acceptance survey appears in Appendix D.

7.1.4 Hypotheses

The summative evaluation tested the following hypotheses. For all hypotheses, a p-value below 0.05 indicated evidence of a statistically significant difference.

Hypothesis H1: IMP will make training more effective than the status quo. Experimental evidence for H1 supports the present dissertation’s summative hypothesis.

Hypothesis H1A: Participants in the Q condition will outperform control participants in the scored simulator scenarios. Related metrics include target selection and engagement scores collected in the two simulator tests.

Hypothesis H1B: Participants in the Q condition will outperform control participants on a test of declarative knowledge. Related metrics include the written test recall, extension, and total scores.
Hypothesis H\textsubscript{1C}: Participants in the Q condition will report more positive perceptions of training than control participants. Participants in the Q condition will also not report undesirable differences such as greater cognitive load or more negative affect. Related metrics include the cognitive load questionnaire, affective state questionnaire, and user acceptance survey.

Hypothesis H\textsubscript{1D}: Participants in the Q condition will demonstrate deeper understanding of the practice material than control participants. Related metrics include target selection and engagement scores in the second simulator test and extension items on the written test of declarative knowledge.

Hypothesis H\textsubscript{2}: Inquiry modeling will increase ITS efficacy. Experimental evidence for H\textsubscript{2} supports the present dissertation’s inquiry modeling hypothesis.

Hypotheses H\textsubscript{2A-D}: These hypotheses and metrics parallel H\textsubscript{1A-D}, but compare condition Q outcomes to participants in condition P instead of the control condition.

Hypothesis H\textsubscript{2E}: IMP models of specific learners will more closely reflect reality in the Q condition than in the P condition. Related metrics are those described for H\textsubscript{3B} and H\textsubscript{3C}.

Hypothesis H\textsubscript{3}: A POMDP learner model will effectively control an ITS. Experimental evidence for H\textsubscript{3} supports the present dissertation’s planning hypothesis. In addition, IMP’s creation and ability to model a realistic military training domain with a POMDP demonstrates support for the dissertation’s scaling hypothesis.
Hypothesis H$_{3A}$: IMP will adapt its interventions and personalize them to individual learners. No statistical comparisons are possible for this hypothesis. Interactions were examined for patterns in order to explore this aspect of IMP.

Hypothesis H$_{3B}$: Estimates in IMP’s learner model will correlate with the actual knowledge gaps learners demonstrate. Related metrics are performance in simulator practice and tests as compared to reports from individual learner models.

Hypothesis H$_{3C}$: Estimates in IMP’s learner model will correlate with the actual affective states learners report. Related metrics are cognitive state questionnaire reports as compared to estimates from individual learner models.

7.2 Results

All hypotheses were tested at the 0.05 confidence level. However, when results reached the 0.01 or 0.001 levels, that additional evidence is reported in this section.

7.2.1 Participants

Before beginning recruitment, a prospective statistical power analysis was performed. The analysis determined the sample size required to ensure an 80% chance to detect effects of a medium size (Cohen’s $f^2 = 0.15$) at the 0.05 confidence level. For three experimental conditions and no covariates, 33 participants per condition were required.

In a previous study using the same simulator and similar training tasks, 25% of participants had to be eliminated from data analysis because of failure to complete any training tasks (Vogel-Walcutt, Gebrim, Bowers, Carper, & Nicholson, 2011). For this reason, up to 132 participants were solicited, with the goal of discarding up to 25% of
participants and still retaining 99 participants. The criterion for elimination from analysis was failure to take any measurable action during both practice sessions. Participants who displayed any measurable performance in either practice session were retained in the analysis, even if they did not perform in the test sessions.

A total of 126 participants were recruited through public postings on the local college campus and Craigslist.org, a popular community website ("Orlando et cetera jobs classifieds," 2011). Out of these participants, data for one participant was lost due to experimenter error and data for 19 participants (15%) were discarded due to the participants’ failure to take any measurable action during both practice sessions. Data analyses therefore included 106 participants. The participants were distributed with 35 in the control condition (O), 37 in the +POMDP condition (P), and 34 in the +POMDP +QUI condition (Q).

Participants’ ages ranged from 18 through 52, with a median age of 22 ($\mu = 24.2$, $\sigma = 6.6$). Males outnumbered females, with 64% of the sample. Most participants (66%) had some college education, while 21% had completed an undergraduate degree or more and 13% had a high-school education or less. Participants reported playing video games for only a few hours per week on average ($\mu = 6.1$, $\sigma = 8.7$; 21% reported no use), but using computers more often ($\mu = 24$, $\sigma = 17$; 0% reported no use). All participants reported at least average comfort with using computers and all reported moderate to strong agreement with the statement “Computers can help me learn difficult course concepts” (Jackson, Graesser, & McNamara, 2009). Finally, no participants had any military training, while five had some knowledge of forward observers and ten recalled
using a military simulator in the past. In none of these cases did experimenters judge a participant’s reported experience sufficient to change the outcome of the experiment.

Participants were randomly assigned to experimental conditions. After the experiment, analyses were performed to ensure that participants with different demographic backgrounds were evenly distributed across experimental conditions. Pearson’s chi-square tests were performed on participants’ gender, educational background, comfort with computers, and confidence in computer trainers. No differences between conditions rose to significance, even without Bonferroni correction for multiple comparisons. To test for imbalance in the continuous demographic variables, Fisher’s least significant difference test was applied without correcting for multiple comparisons. Still, no difference between conditions was detected in participants’ reported age, computer usage, or video game usage.

An apparently high number of dropped participants were in the Q condition: 11 as compared to four in each of the O and P conditions. Therefore, a chi-square test was performed to determine whether dropping a participant was related to their experimental condition. The test did not reject the null hypothesis, so the participant drop rate was statistically independent of experimental condition ($\chi^2 = 4.666$, d.f. = 2, n.s.).

Since experimental condition did not determine which participants were dropped, an alternative explanation was sought. Examination of several demographic variables showed participants’ reported age predicted whether they would drop from the study with high sensitivity and specificity. F$_1$ scores, a combination of sensitivity and specificity varying between 0 and 1 (for perfect predictions), reached 0.80 for reported age
compared to a maximum of 0.63 when using experimental condition as a predictor. The age threshold that maximized $F_1$ fell between 25 and 26. Participants older than this threshold accounted for 33% of the pooled sample, but 84% of the dropped participants.

As a direct comparison with the above test on experimental condition, participants were separated into two groups with reported ages above and below the threshold and an additional chi-square test was performed relating these groups to the study dropout rate. Unlike the experimental condition, reported age was indeed significantly related to dropping from the study ($\chi^2 = 26.867, \text{d.f.} = 1, p < 0.001$). Furthermore, because of random assignment half of all participants in the older group were in the Q condition while only a quarter were in each of the O and P conditions. While the difference between experimental condition assignments was not significant ($\chi^2 = 4.457, \text{d.f.} = 1, n.s.$), it helped explain why the Q condition had more dropped participants than the other two conditions.

### 7.2.2 Proficiency Tests

Two proficiency evaluations measured how well participants demonstrated target skills in a performance environment, as opposed to declarative knowledge on a written test. The first test evaluated proficiency in the same scenario as the participants had practiced, while the second test evaluated proficiency in a related scenario that the participants had not seen before. In each of the two tests, outcomes reported how much of the scenario participants were able to complete, how well they selected targets to attack, and how well they followed the rules for engaging each target. Shapiro-Wilk tests for
normality showed that proficiency outcomes were not normally distributed, so non-parametric Kruskal-Wallis one-way ANOVAs were performed to detect differences between conditions.

As Figure 21 shows, no significant differences between conditions were found in the number of calls for fire issued, the number of enemies destroyed, or the number of fratricides. This finding supports the hypothesis that added interventions from the tutors in the P and Q conditions would not significantly interrupt training or otherwise decrease participants’ automaticity and speed in calling for fire.

![Simulator Recall Test Completion](image1)

![Simulator Extension Test Completion](image2)

*Figure 21: For both simulator tests, participants in different conditions did not differ significantly in their speed of issuing CFFs or in their rate of hitting enemy targets or friend targets. Therefore any tutor interruptions during practice did not cause a significant impact on automaticity or proficiency in using the CFF simulator. Error bars indicate 95% confidence intervals.*

On the recall proficiency test, differences were found between conditions in target selection scores ($H = 12.91$, d.f. = 2, $p < 0.01$). Target selection outcomes are shown in Figure 22. Post-hoc tests for least significant difference of mean ranks showed that participants in the P and Q conditions performed significantly better than the control condition ($O-P \Delta = 16.74$, $p < 0.01$; $O-Q \Delta = 25.11$, $p < 0.001$). Participants in the P
condition outscored the control participants on average by 0.64 standard deviations. In the Q condition, the improvement was 0.94 standard deviations. While Q participants did have better scores than P participants, the difference between conditions was not significant ($\Delta = 8.38$, n.s.).

More significant differences were found between conditions in scores for recall of target engagement ($H = 22.93$, d.f. = 2, $p < 0.001$). Post-hoc tests showed that all differences between conditions were significant. Participants in the P condition again outperformed control participants by 0.64 standard deviations ($\Delta = 13.38$, $p < 0.05$), and in turn participants in the Q condition outperformed P participants by 0.82 standard deviations ($\Delta = 18.78$, $p < 0.01$). Participants in the Q condition also outperformed control participants ($\Delta = 32.17$, $p < 0.001$). In the O-Q comparison, the performance improvement amounted to 1.77 standard deviations. This measure represented a particularly large difference between conditions, with participants using IMP scoring 4.5 times higher than participants in the control condition and twice as high as those who used ablated IMP with no inquiry modeling functionality.
Figure 22: Participants in the P and Q conditions demonstrated significantly better target selection than the control participants. In target engagement, all conditions were significantly different, with P better than the control and Q better than P. Error bars indicate 95% confidence intervals.

For the extension tests of proficiency, no outcomes were significantly different across conditions. The measures that most closely approached significance were number of enemies hit and number of fratricides. These two measures showed a trend that P outscored O and in turn Q outscored P, but only the improvements in fratricide count
reached even marginal significance \((p = 0.09, n.s.)\). Target selection and target engagement scores in the extension test were close to identical across all conditions.

### 7.2.3 Written Knowledge Tests

A written test of knowledge measured how well participants were able to recall and apply knowledge outside of a simulator setting. Shapiro-Wilk tests for normality and Levene’s tests for homoscedasticity did not detect any problematic distributions, so outcome differences were checked with parametric linear regression. Participants in different conditions accomplished significantly different outcomes on the written test \((F = 9.494, \text{d.f.} = (2, 103), p < 0.001)\). Post hoc tests using Tukey’s honestly significant difference (HSD) method showed that participants in the Q condition outscored both control participants \((p < 0.01)\) and participants in the P condition \((p < 0.01)\). The mean improvement amounted to 0.83 and 0.85 times the pooled standard deviation, respectively. Participants in the P condition scored fractionally less than the control participants, although the score difference was not significant. Figure 23 compares the groups’ scores.

About half of the written test items targeted direct recall of knowledge that was taught before the practice, while the other half required participants to apply that knowledge in new situations that had not appeared during practice. These item groups were next analyzed separately. Experimental condition strongly affected the recall items \((F = 17.586, \text{d.f.} = (2, 103), p < 0.001)\). Participants in the Q condition outscored control participants by 1.2 standard deviations \((p < 0.001)\), and outscored P participants by 0.95
standard deviations ($p < 0.001$). This time participants in the P condition did outscore control participants, but the difference was again not significant. Experimental condition did not appear to affect knowledge application ($F = 1.439$, d.f. = (2, 103), n.s.). In sum, most of the difference in the written measure outcome was attributable to a large improvement in recall ability, but without a significant improvement in ability to apply the learned knowledge on the written extension questions.

![Written Test Scores](image)

Figure 23: Participants in the Q condition significantly outscored participants in the P condition and control participants, who scored about the same. Although there was no difference between conditions on extension test items requiring application to new situations, the Q participants showed significantly improved recall of knowledge directly related to the practice. Error bars indicate 95% confidence intervals.

7.2.4 User Acceptance

Each of the 24 user acceptance survey items were evaluated separately. A five-factor model had been posited before the experiment to relate certain items into groups,
but after the experiment a confirmatory factor analysis did not find support for the proposed model in the experimental data. The sample size of the study may have been too small to permit such an analysis.

Survey items were analyzed by finding the percentage of participants who agreed or strongly agreed with each statement, or disagreed and strongly disagreed with reversed items. Pearson chi-square tests found significant differences between conditions in seven items, and two more items showed marginally significant differences. Participant responses to these nine items only are shown in Figure 24. Notably, there were no significant differences in any of the four survey items measuring task value or perception of the training components outside of IMP, such as the expository videos.

On three of the four survey questions relating to self-efficacy, significantly more participants responded positively in the Q condition than in the other two conditions. These participants were more likely to feel that they remembered the KSAs from the training ($\chi^2 = 6.690$, d.f. = 2, $p < 0.05$), that they could apply these KSAs in new situations ($\chi^2 = 6.427$, d.f. = 2, $p < 0.05$), and that they got high test scores ($\chi^2 = 12.888$, d.f. = 2, $p < 0.01$). On the final self-efficacy question, satisfaction with their test performance, the difference only trended towards statistical significance, but again in favor of the Q condition ($\chi^2 = 5.793$, d.f. = 2, $p = 0.055$, n.s.). Finally, participants in the Q condition were also significantly more likely to agree that IMP provided encouragement when they needed it ($\chi^2 = 7.426$, d.f. = 2, $p < 0.05$).

Participants in the control condition were significantly more likely than the other participants to state that the tutor did not distract them ($\chi^2 = 18.665$, d.f. = 2, $p < 0.001$)
and marginally more likely to state that the tutor did not tell them things they already knew ($\chi^2 = 5.434$, d.f. = 2, $p = 0.066$, n.s.). Since the tutor in the control condition did not tell participants anything, these responses are not surprising.

Interestingly, participants in the P condition were both significantly more likely to judge IMP’s hints as helpful ($\chi^2 = 7.758$, d.f. = 2, $p < 0.05$) and significantly less likely to agree that they would like to use IMP again ($\chi^2 = 7.563$, d.f. = 2, $p < 0.05$).
a01: I can remember the knowledge and skills from the teaching videos and the practices.

a02: I feel ready to apply the knowledge and skills from the videos and practices in new situations.

a23: I got high scores on the tests.
a24: I felt satisfied with my performance on the tests. (marginally significant)

a16: The practice trainer encouraged me when I needed it.

a15: During practice, the trainer distracted me from working or remembering. (reversed)

a13: The practice trainer told me things I already knew. (reversed, marginally significant)
a12: The practice trainer gave me useful hints or reminders.

![Bar chart showing participant responses to item a12](chart1)

a18: I would like to learn more from this particular practice trainer.

![Bar chart showing participant responses to item a18](chart2)

Figure 24: Nine user acceptance survey items that showed the most significant differences between conditions in the number of participants who agreed or strongly agreed. Items are ordered by their introduction in the text. For each item, the three horizontal bars correspond to the three experimental conditions, and are divided according to the distribution of participant responses. Colored bars highlight the percentage of participants who agreed and strongly agreed with each item (or disagreed and strongly disagreed with reversed items).

7.2.5 Tutor Interaction Patterns

IMP’s interactions with participants in the P and Q conditions were examined for patterns. Repeating the same actions for many participants would indicate IMP was not responsive to trainees’ needs. Figure 25 and Figure 26 display all sequences of tutor actions that occurred more than once over all participants. In these figures, rectangle nodes correspond to an intervention IMP initiated. Oval nodes indicate one or more intervention sequences that only occurred for a single participant. Learner-initiated help
requests are not included in these figures; neither are recommendations that were not implemented.

Examining IMP’s intervention choices demonstrates that for almost 90% of participants, the sequence of hints IMP displayed was not the same as any other participant saw after only five turns (Figure 27). IMP’s adaptive behaviors were responsive to the performance and the training needs of each individual participant.

In both experimental conditions, IMP’s initial action before any input from the trainee was to present information about how to use the simulated rangefinder, labeled $RF$ in Figure 25 and Figure 26. From this starting point IMP successfully moved through the state queue in either direction, presenting more remedial interventions such as the one indicated by the node $Menu$ or more advanced information such as $Friend 1$. By the time a second intervention was called for, IMP was ready to recommend a variety of different interventions according to the needs of each participant.

In general, 60% of tutor recommendations during practice did not directly address the estimated most-probable gap. IMP did not merely follow a greedy strategy to select its actions. IMP’s selections also were not simply reactive, without memory or planning. Considering only recommendations IMP made immediately after observing one or more trainee errors, 51% did not directly address any of the errors just observed. These two patterns support the assertion that IMP planned ahead and personalized interventions, rather than using a more obvious intervention strategy.

Neither the P version nor the Q version of IMP took advantage of interventions with solely affective outcomes. They chose to present only interventions that had a
chance to clear some knowledge gap. IMP also selected “no-op” actions rarely, and only at the end of a training session. IMP did not believe in a “wait-and-see” approach to determining what training would help a participant. The Q version of IMP also never employed its low-distraction interventions that highlighted questions for a user in the QUI rather than immediately showing information.

Finally, there were qualitative differences between the tutoring choices of the P and Q versions of IMP. The Q version presented hints on more advanced material earlier than the P version, and the Q version showed greater diversity of hints selected. The differences may be due to trainees asking the Q version questions about the simpler material, while the P version needed to use some of its few tutor-initiated interactions to show simpler material. The Q version was less likely to present the hint Menu about using the radio menu, and it was more likely to present the hint Hurry. Menu is a recommendation corresponding to the first gap in the state queue, while Hurry corresponds to the last gap, indicating a trainee thinks the training task is complete when actually enemy targets remain. For a certain group of trainees, IMP selected this intervention and urged them to quickly attack a new target several times in a row.
Figure 25: Hints that IMP chose for more than one participant in the P condition. Oval leaves represent several diverging choices that were different for every participant (label indicates number of paths). By the fifth turn, IMP’s personalization was responsive enough to present different material to every participant except for two pairs of people.
Figure 26: Hints that IMP chose for more than one participant in the Q condition. Oval leaves represent several diverging choices that were different for every participant (label indicates number of paths). Tutor interactions diverged more quickly than in the P condition, suggesting learner questions offered more observations to help IMP personalize its interactions. However, one group of participants received the “Hurry” hint several times in a row.
Figure 27: In both experimental conditions, IMP presented most participants with unique interventions rather than interventions that it showed to any other participant. In the Q condition, IMP’s decision paths diverged more quickly than in the P condition.

7.2.6 Tutor Help Efficacy

Statistical tests were conducted to examine how different IMP components affected participant outcomes. Results of this nature are related to certain POMDP parameters that were set by hand for the present experiment. In future work, simple statistics like these could inform the design process to make POMDP learner models more accurate or to lighten the development workload on subject-matter experts.

Pearson correlation tests compared written measure outcomes with the number of IMP-selected hints displayed to participants in the P and Q conditions, and the number of help requests from participants in the Q condition. The number of hints IMP interjected
correlated with participants’ scores on the written test overall ($\rho = 0.30$, two-tailed $p < 0.05$) and specifically on the extension items ($\rho = 0.35$, two-tailed $p < 0.01$), but was not related to performance on the recall items. Considering only participants in the Q condition, participants’ asking more questions correlated with decreased performance on the extension items of the written test ($\rho = -0.40$, two-tailed $p < 0.05$), but was not related to the recall or total scores.

Just as the experimental conditions did not slow down completion rates in the two simulator tests (Section 7.2.2), IMP’s interventions were also unrelated to completion speed. The sole exception was that the number of hints IMP chose to display was closely related to the number of CFFs participants completed in the first ($\rho = 0.67$, two-tailed $p < 0.001$) and second tests ($\rho = 0.60$, two-tailed $p < 0.001$). However, this difference entirely disappeared after controlling for the number of CFFs the participants issued during practice (first test $\rho = -0.06$, n.s., second test $\rho = 0.03$, n.s.). This indicated that the correlation was not due to the hints changing test speed; rather, hint count and test speed both varied with practice speed.

Considering a finer level of detail, correlations were calculated between the simulator test outcomes and the number of IMP interventions that specifically targeted those tests. The number of target engagement hints participants read was significantly correlated with improved engagement scores in both the simulator recall test ($\rho = 0.40$, $p < 0.01$) and also the simulator extension test ($\rho = 0.32$, $p < 0.05$), where experimental condition alone had not led to any score difference (see Section 7.2.2). For target
selection, no correlation appeared between the number of hints participants read and their scores in the simulator tests (first test $\rho = 0.02$, n.s., second test $\rho = 0.18$, n.s.).

Finally, participants’ demographic backgrounds were considered to find any patterns in their likeliness to ask for help. In the present sample, and considering only participants in the Q condition, no demographic items were significantly correlated with the number of help requests.

### 7.2.7 Tutor Belief Accuracy

IMP’s learner model was evaluated to determine how closely the models reflected trainees’ actual cognition. First, trainees’ knowledge and proficiency outcomes were compared to IMP’s estimates of their knowledge state at a single point in time, the end of practice. Second, the affective state estimates in IMP’s learner model throughout training and testing were compared to trainees’ recall of their affect while using the simulator.

#### 7.2.7.1 Modeling trainee knowledge

First, for each participant, IMP’s maximum a posteriori (MAP) estimate of knowledge after completing training was examined. MAP estimates reduce a belief distribution to a single point, reporting which model state has the highest estimated probability conditioned on the observed evidence.

For most participants, IMP’s belief distribution at the end of training was visibly unimodal. For about 82% of participants across all three conditions, the most probable knowledge state was the final state in the queue, representing no knowledge gaps remaining. Furthermore, the estimated marginal probability of participants being in the
final knowledge state averaged 73%, with many individual estimates approaching 100%. Based on subsequent performance evaluations, these estimates were too optimistic.

Since knowledge estimates clustered around the same state for many participants, it was difficult to find linear relationships between IMP’s knowledge estimates and actual performance. Instead, participants were grouped into two bins, comparing those whom IMP estimated had knowledge gaps at the end of training with those whom IMP estimated had none. IMP’s estimates correctly reflected participants’ performance during practice. During practice, the better-rated group completed almost twice as many practice CFFs ($p < 0.001$), scored significantly better on target engagement ($p < 0.05$), and scored marginally better on target selection ($p = 0.10$, n.s.). After practice ended, participants to whom IMP ascribed no knowledge gaps performed significantly better on the written test ($p < 0.05$). The difference was large: 2.8 times the standard deviation. For simulator task proficiency scores, however, non-parametric Mann-Whitney U tests did not reveal significant differences between the two groups.
Figure 28: Performance during practice of the participants IMP estimated were done at the end of practice (top group), compared to those IMP estimated still had knowledge gaps at the end of practice (bottom group). IMP's knowledge estimates correctly reflected significant differences between groups during practice. However, the model did not distinguish between learners in a finer gradation than "Done" or "Not Done."

7.2.7.2 Modeling trainee affect

In evaluating IMP’s affective modeling, ground truth was determined after each simulator session in terms of what fraction of the time participants experienced each modeled affective state. The four affective states of boredom, confusion, flow, and frustration were considered separately. In order to reduce inter-participant variability, responses about a particular state were aggregated into three groups composed of those who rarely or never experienced the state, those who reported experiencing the state much of the time or always, and those who gave intermediate responses. Pearson chi-square tests did not reveal differences between experimental conditions in trainees’ reported affective states or cognitive load perception.

For each simulator session and affective state, IMP’s learner model reported the percentage of practice time when the MAP-estimate state shared the same affect. Since IMP’s model allowed for only one affective state at a time, while participants could report multiple states, the model reports were not expected to reach 100% agreement even when participants actually experienced some affective state for the entire session. Instead, a successful model should put the three groups in increasing frequency order and with statistically significant differences between each group’s frequency.

Table 22 records the results of ANOVA and Tukey’s HSD post-hoc tests on the four affective states modeled during the four simulator sessions. As Table 22 shows, in eight out of sixteen cases IMP ascribed the target affective state to participants in the
three groups with frequency different enough to be statistically significant. In all eight cases, the direction of the difference was correct, as opposed to wrongly detecting a state less often even though participants reported it more frequently. The eight cases with significant differences are shown in Figure 29. In the other eight cases, IMP did not place participants in the target affective state any more or less often across the three groups. In some cases, this failure to find a significant difference may have been due to group size imbalance, since participants reported boredom, confusion, and frustration rarely.

The IMP estimations aligned with participant reports on three of the four states modeled during both the first practice and the first simulator test, only failing to discriminate different levels of boredom. However, IMP failed to produce significant differences between groups in any state during the second simulator practice. During the second simulator test, IMP’s beliefs aligned with participant reports of boredom and flow, but not with their confusion or frustration.

Table 22: Results of testing whether IMP’s affective state estimates differed between groups of participants who actually felt the same state more or less frequently. Eight of sixteen differences were statistically significant. Post-hoc tests (third line in cells with significant differences) show that where differences existed, they were in the correct direction.

<table>
<thead>
<tr>
<th></th>
<th>Boredom</th>
<th>Confusion</th>
<th>Flow</th>
<th>Frustration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simulator Practice</strong></td>
<td>$F_{(2,97)} = 0.577$ n.s.</td>
<td>$F_{(2,98)} = 6.553$ $p &lt; 0.01$ lo &lt; hi</td>
<td>$F_{(2,97)} = 5.644$ $p &lt; 0.01$ lo &lt; hi, mid &lt; hi</td>
<td>$F_{(2,98)} = 9.936$ $p &lt; 0.001$ lo &lt; mid, lo &lt; hi</td>
</tr>
<tr>
<td><strong>Domain Practice</strong></td>
<td>$F_{(2,98)} = 0.419$ n.s.</td>
<td>$F_{(2,98)} = 0.805$ n.s.</td>
<td>$F_{(2,98)} = 1.196$ n.s.</td>
<td>$F_{(2,98)} = 1.822$ n.s.</td>
</tr>
<tr>
<td><strong>Recall Test</strong></td>
<td>$F_{(2,98)} = 0.026$ n.s.</td>
<td>$F_{(2,98)} = 5.099$ $p &lt; 0.01$ lo &lt; mid</td>
<td>$F_{(2,98)} = 6.471$ $p &lt; 0.01$ mid &lt; hi</td>
<td>$F_{(2,98)} = 4.837$ $p &lt; 0.05$ lo &lt; hi, mid &lt; hi</td>
</tr>
<tr>
<td><strong>Extension Test</strong></td>
<td>$F_{(2,98)} = 3.819$ $p &lt; 0.05$ lo &lt; mid</td>
<td>$F_{(2,98)} = 1.198$ n.s.</td>
<td>$F_{(2,98)} = 8.036$ $p &lt; 0.01$ lo &lt; mid, lo &lt; hi</td>
<td>$F_{(2,98)} = 1.046$ n.s.</td>
</tr>
</tbody>
</table>
198

Figure 29: Participant retrospection of affective states was compared to IMP estimates. In all cases where IMP estimates differed between groups, shown here, the direction of the difference was correct. Black markers indicate significant differences, while light grey markers are shown for completeness. Error bars indicate 95% confidence intervals, and have horizontal width proportional to the number of participants in each group.

7.2.8 Summary of Results

This section presented results from an empirical evaluation of the Inquiry Modeling POMDP adaptive trainer (IMP). Table 23 summarizes the study’s main findings. IMP produced significant improvements in several measures of knowledge and
skill in the target domain. For some of the improvements, the differences between experimental conditions were quite large. The full version of IMP produced better improvements than an ablated version with no interface for trainees to request help from the trainer. For some measures that did not show an improvement across all IMP users, performance improvements still correlated with feature usage, so participants who used IMP more accomplished better learning outcomes. IMP’s extra interventions did not significantly decrease participants’ skill automaticity or proficiency. Finally, the POMDP learner model was found to broadly align with participants’ performance during training and even reflected some of their affective reports accurately. The model was able to support diverse and responsive adaptations of IMP’s interventions to each trainee’s needs.
Table 23: Overall, the study showed that IMP improved training. Inquiry modeling produced better results than an ablated IMP without inquiry modeling. The POMDP learner model reflected trainees’ knowledge and affective states.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Hypothesis</th>
<th>Supported $(p &lt; 0.05)$</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMP vs. status quo</td>
<td>Simulator performance ↑ Yes Scores improved by up to 1.77σ.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Declarative knowledge ↑ Yes Mean score improved by 0.83σ.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User perception ↑ Yes Better on five measures, worse on one.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deep understanding ↑ No However, those who saw more hints did</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>show improved deep understanding.</td>
<td></td>
</tr>
<tr>
<td>Inquiry-modeling IMP vs. ablated</td>
<td>Simulator performance ↑ Yes Scores improved by up to 0.82σ.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Declarative knowledge ↑ Yes Mean score improved by 0.85σ.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>User perception ↑ Yes Better on five measures, worse on one.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Model accuracy ↑ No No significant differences in model accuracy measures.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>POMDP learner model</th>
<th>Interventions personalized</th>
<th>Not Applicable</th>
<th>The first five interventions were different for 90% of participants.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test scores confirm knowledge gap model</td>
<td>Yes</td>
<td>Participants IMP estimated to have no gaps outperformed others by up to 2.8σ.</td>
<td></td>
</tr>
<tr>
<td>Questionnaires confirm affective state model</td>
<td>Yes</td>
<td>Eight of 16 cases showed significant agreement; none significantly disagreed.</td>
<td></td>
</tr>
</tbody>
</table>

7.3 Discussion

The previous section reported statistical results of evaluating IMP. The statistics showed that, overall, IMP improved knowledge recall and skill performance outcomes. The present section looks more closely at questions about POMDPs for modeling and planning in IMP and about inquiry modeling as an additional learner interaction channel. It examines IMP’s interactions with selected individual learners to point out what went right and where there existed opportunities for improvement. Sections 7.3.1 and 7.3.2 focus on IMP’s POMDP learner model, and Sections 7.3.3 and 7.3.4 discuss the inquiry modeling paradigm.
7.3.1 POMDPs

The statistical analyses in Section 7.2.5 demonstrated that in general, IMP did not employ a simplistic policy such as reacting to a learner’s last error or tutoring the most likely knowledge gap in the learner model. The session transcript in Table 24 displays typical IMP interactions with one learner in the P condition (ablated IMP with no learner-initiated questions). For an explanation of the error and action names, see Section 6.3. The present section interprets each of the session’s interactions in detail to show an anecdotal example of IMP’s POMDP learner model and complex interaction policy in action.

In the Table 24 transcript, IMP had only nine opportunities to initiate an interaction with the learner before practice ended. During this time, the learner’s observable performance included nine different errors, unevenly distributed so that in three cases the learner demonstrated several errors at once and in one case the learner performed without any errors. As was common throughout this experiment, IMP did not have time to address all the errors demonstrated, but its actions still formed a coherent instructional pattern.

First, IMP had an opportunity to present some intervention at the start of practice, before observing any trainee performance. IMP’s recommendation was always the same at this point in training, presenting the hint RF which reminded trainees how to use the simulated rangefinder tool effectively. The RF hint was an effective choice to start training, and not necessarily an obvious one. Most trainee actions in the present experiment required correct rangefinder use because incorrect use could lead not only to
shots that miss an enemy but to the simulator misinterpreting the trainee’s intended target and therefore sending IMP incorrect performance assessments. However, RF was not the first knowledge gap in IMP’s state queue. In fact, according to IMP’s model, tutoring relied first on three other, more basic gaps. IMP chose not to start by addressing those other gaps, probably because the learner model estimated they were less likely to be present in the initial learner state. Furthermore, IMP also did not simply address the most likely gap. One other gap, Priority1, was modeled as more likely to be present \textit{a priori} than RF. But IMP chose to tutor RF before Priority1 because the dependencies reflected in the state queue made tutoring RF likely to assist in tutoring more gaps later, including the Priority1 gap. Therefore, IMP’s first default action demonstrated planning ahead for effective training.

In line 1 of Table 24, the participant completed a call for fire with no errors. After this observation, IMP recommended that an effective intervention would be the Friend1 hint, reminding the trainee not to attack one of the two friendly targets in the simulator. This made sense because even though the trainee had not attacked a friendly target or made any other error, trainees were expected \textit{a priori} to be confused at the start of training about differentiating friendly and enemy targets. However, the first practice session focused only on familiarization with the simulator, and trainees were not instructed on target differentiation until after the first practice. Therefore, the Friend1 recommendation was overridden by a rule outside of the POMDP (Section 6.3.3) and IMP did not get another chance to recommend an intervention on that turn. The first practice session ended with no more interactions, and on line 2 the second practice
session started. Before the second practice session trainees learned about differentiating friends from foe targets, so IMP displayed the *Friend1* hint at the start of practice.

On Line 3, the trainee completed a call for fire that had several errors. However, IMP did not immediately address any of the errors that were demonstrated. Instead, IMP chose to address the *Foe2* knowledge gap from higher in the state queue (a more basic knowledge state), which the model predicted would help make addressing the observed errors later more effective. Furthermore, *Foe2* had a possibility of causing the prioritization error the trainee had just committed. Like *Friend1* from the previous turn, *Foe2* had a high initial probability of being present at the start of training. These considerations made IMP’s choice an effective one.

IMP next had a chance to act on Line 4 when the trainee failed to send a call for fire in the required time period. IMP used this opportunity to directly address the *Priority1* error that it had observed on the previous turn. On Line 5, IMP continued to address target prioritization with a second hint, *Priority2*. Even though IMP had not observed the trainee commit the error corresponding to *Priority2*, the observation of the previous prioritization error had made the presence of this knowledge gap more likely.

With the next call for fire on Line 6, the trainee somewhat vindicated IMP’s assumption that the *Priority2* gap existed by committing the error that IMP had just addressed. However, IMP did not repeat itself to address the error a second time. Instead, IMP addressed the next-most advanced error out of those the trainee had demonstrated. The decision to move on to training more advanced topics rather than dwelling on the basic topic made sense if the trainee’s mistake in basic target selection was merely a slip.
On Line 7, the trainee sent a call for fire attacking a friendly target. When IMP observed such a basic error for a second time, it recontextualized the previous evidence. Now, rather than a slip, IMP decided that the trainee really did need more help on basic material. Therefore IMP presented the *Priority2* hint from which it had forborne in the previous turn. However, IMP did not directly address the trainee’s error in attacking a friendly target, which it probably should have. Since the error had been observed twice at this point, it was probably evidence of a basic knowledge gap that IMP should have addressed, rather than a slip that it should have ignored. Indeed, the trainee did go on to attack the same type of friendly target again during the performance test.

IMP’s final opportunity to intervene came on Line 8, after the trainee sent one more call for fire with several errors. In this case, IMP observed three errors and had to choose which one to address. IMP chose to address *Ammo1*, the most advanced of the three errors, that is, the one with the lowest priority in IMP’s state queue. IMP probably made this choice because it had not observed the more basic errors before, but it had observed the trainee make the *Ammo1* error once already. Therefore, it might make sense in such a case to address a more advanced topic that is likely to be a real knowledge gap before a more basic topic that could be a slip. Although training ended at this point, IMP had already shown in this transcript that if needed, it was capable of backing up to address a basic point after addressing an advanced gap.

After the practice session described in the present section, the trainee’s target engagement improved on the performance tests as compared to practice and was above the group average. However, the trainee’s target selection scores decreased compared to
practice and fell all the way to zero. The trainee attacked four friendly targets during the two tests, indicating that IMP’s failure to address the friendly-fire mistake when it appeared during practice probably hurt the trainee’s target selection performance.

*Table 24:* A transcript of a typical training session in the P condition demonstrates how the POMDP learner model lets IMP address likely knowledge gaps before it directly observes them, choose which gap to address when it observes several, ignore mistakes that may be slips, and generally intervene more flexibly than a decision tree would.

<table>
<thead>
<tr>
<th>Trainee Action</th>
<th>CFF Errors</th>
<th>IMP Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of Simulator Practice</td>
<td>No Errors</td>
<td>RF</td>
<td>RF was recommended based on the initial belief, before any observations.</td>
</tr>
<tr>
<td>1. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td>Friendship refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>2. Start of Domain Practice</td>
<td>Friend1</td>
<td></td>
<td>The previously recommended Friendship action was deferred to the first allowed time, the start of the next practice.</td>
</tr>
<tr>
<td>3. CFF</td>
<td>Priority1</td>
<td>Foe2</td>
<td>Though the Foe2 gap was not observed, Foe2 can account for seeing Priority1, and clearing the observed gaps depends in the model on clearing Foe2 first.</td>
</tr>
<tr>
<td>4. Long Idle</td>
<td>Priority1</td>
<td></td>
<td>Priority1 was the highest priority gap observed so far.</td>
</tr>
<tr>
<td>5. CFF</td>
<td>Friend2</td>
<td>Priority2</td>
<td>The Priority2 gap was not observed, but observing Priority1 made it likely.</td>
</tr>
<tr>
<td>6. CFF</td>
<td>Priority2</td>
<td>Desc2</td>
<td>IMP believed the higher-priority error Priority2, observed here after that gap was addressed, was merely a slip. It addressed the lower-priority Desc2.</td>
</tr>
<tr>
<td>7. CFF</td>
<td>Friend2</td>
<td>Priority2</td>
<td>Observing the basic error Friend2 made IMP believe the previous mistakes were gaps, not slips. IMP backed up to address the gap Priority2 from last turn, but failed to directly address Friend2.</td>
</tr>
<tr>
<td>8. CFF</td>
<td>Fire1</td>
<td>Ammo1</td>
<td>The first time IMP observed Ammo1 it needed to address another gap, but now IMP addressed it after seeing it again.</td>
</tr>
</tbody>
</table>
7.3.2  POMDP Lessons Learned

There were three areas in which anecdotal evidence from individual interaction logs showed IMP’s POMDP learner model left room for improvement. The model was too optimistic in interpreting error observations as slips, could have benefited from the inclusion of user demographic information, and failed to use the full range of actions available to it.

7.3.2.1  Interpret more slips as errors

First, the typical training session described in the previous section demonstrated one instance when IMP’s optimism was not justified. IMP saw the trainee attack a friendly target twice without correcting the trainee. IMP should have addressed the knowledge gap underlying this error unless the incorrect performance was in fact merely a slip. Examining several participants’ records revealed that while IMP corrected friendly-fire mistakes when they happened early in a practice, IMP did not correct trainees who attacked a friendly target after first attacking several enemy targets. In IMP’s learner model, attacking an enemy serves as evidence that the learner can identify both enemy targets and friendly targets. IMP’s behavior pattern suggests that when a trainee has attacked some enemy targets, IMP discounts later observations of friendly fire during practice as slips.

In other trainee sessions, IMP was overly optimistic about more gaps than just the ones related to friendly fire. Table 25 transcribes a practice session in the P condition when IMP had 18 intervention opportunities—more than almost all participants and twice
as many as the average participant. This unusually long path through the learner model revealed the weakness of IMP’s optimism about several knowledge gaps because IMP did not effectively use four of its last six opportunities to initiate interactions. Even though the participant made several errors during the session, there were enough correct performances on each point that by the end of the second practice IMP believed no knowledge gaps remained and did not address observed mistakes, instead repeating the *Hurry* hint. Coincidentally, the transcript in Table 25 also includes IMP ignoring evidence of the *Friend2* knowledge gap.

IMP’s optimism in general was probably due to its POMDP’s observation parameters, rather than action parameters. Even in the control condition, when IMP did not control any actions, IMP still believed the mean marginal probability that participants had no knowledge gaps at the end of practice was 70% (compared to 73% in the P condition). This suggests the optimism came from assigning misleading weights in the model to some performance observations. Therefore, to correct IMP’s optimism, POMDP observation emission probability parameters should be changed to strengthen the evidence imputed to observations of errors during practice. Weakening the evidence of correct behavior would have the unintended consequence of making IMP present more hints, leading to frustration when the hints are not needed, based on its initial beliefs before any observations. In the ideal case to address this issue, if sufficient supporting data are available, POMDP parameters should be based on real data from previous trainees’ practice.
Table 25: A transcript from a session with 18 tutor turns, more than usual, again shows that IMP planned ahead to tutor effectively. However IMP too easily dismissed evidence of some gaps as mere slips, concluding by the end of the long session that the trainee did not need further help and ineffectually using four of its last six turns.

<table>
<thead>
<tr>
<th>Trainee Action</th>
<th>CFF Errors</th>
<th>IMP Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of Simulator Practice</td>
<td>No Errors</td>
<td>RF</td>
<td><em>RF</em> was recommended based on the initial belief, before any observations.</td>
</tr>
<tr>
<td>1. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td><em>Friend1</em> refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>2. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td><em>Friend1</em> refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>3. Start of Domain Practice</td>
<td>Friend1</td>
<td></td>
<td>The previously recommended <em>Friend1</em> action was deferred to the first allowed time, the start of the next practice.</td>
</tr>
<tr>
<td>4. CFF</td>
<td>Priority1 Desc1 Ammo1</td>
<td>Friend2</td>
<td>IMP addresses the first mistake seen.</td>
</tr>
<tr>
<td>5. CFF</td>
<td>Priority1 Desc1 Ammo1</td>
<td>Priority1</td>
<td>IMP sees several mistakes and addresses the highest-priority gap, <em>Priority1</em>, immediately.</td>
</tr>
<tr>
<td>6. CFF</td>
<td>Fire1 Desc1 Ammo1</td>
<td>Fire1</td>
<td><em>Desc1</em> and <em>Ammo1</em> are repeated, but IMP addresses <em>Fire1</em> first. The other two gaps, again, depend on this one.</td>
</tr>
<tr>
<td>7. CFF</td>
<td>Priority1 Priority2 Desc1 Ammo1</td>
<td>Foe2</td>
<td>IMP guesses that the multiple prioritization errors seen so far are because the trainee is only targeting one type of enemy and may have forgotten which other targets are enemies.</td>
</tr>
<tr>
<td>8. CFF</td>
<td>Friend2 Desc1</td>
<td></td>
<td>IMP previously corrected the <em>Friend2</em> gap, so it believes this mistake is a slip. Instead of correcting it, IMP addresses the <em>Desc1</em> mistake from a previous turn.</td>
</tr>
<tr>
<td>9. CFF</td>
<td>Desc2</td>
<td>Desc2</td>
<td>IMP addresses a gap when it is seen.</td>
</tr>
<tr>
<td>10. CFF</td>
<td>Priority1 Priority2 Desc2</td>
<td>Desc2</td>
<td>IMP attempts to clear the same <em>Desc2</em> gap when it is observed again. IMP believes the priority mistakes are slips.</td>
</tr>
<tr>
<td>11. CFF</td>
<td>Ammo1</td>
<td>Ammo1</td>
<td>The <em>Ammo1</em> gap had been observed before, but higher-priority gaps had to be cleared before IMP could address it.</td>
</tr>
<tr>
<td>12. Shot Miss</td>
<td></td>
<td>Hurry</td>
<td>IMP believes all other gaps are clear.</td>
</tr>
<tr>
<td>13. CFF</td>
<td>No Errors</td>
<td>Ammo2</td>
<td><em>Ammo2</em> had never been observed, but observing <em>Ammo1</em> made it more likely.</td>
</tr>
<tr>
<td>15. Shot Miss</td>
<td></td>
<td>Hurry</td>
<td>IMP believes all other gaps are clear.</td>
</tr>
<tr>
<td>16. CFF</td>
<td>Priority1 Desc2</td>
<td>Desc2</td>
<td>IMP believes <em>Priority1</em> was merely a slip but <em>Desc2</em> was evidence of a gap.</td>
</tr>
<tr>
<td>17. Shot Miss</td>
<td></td>
<td>Hurry</td>
<td>IMP believes all other gaps are clear.</td>
</tr>
</tbody>
</table>
7.3.2.2 *Model some demographic information*

The second change suggested by the present results indicated IMP could have benefitted from demographic information about participants. IMP used the same initial beliefs for all trainees in the present experiment, but the experimental results showed a highly significant difference between older and younger participants in failure to complete enough measurable simulator performance to be included in the analysis. Based on this fact, simulator performance test metrics were next checked for differential outcomes according to the same age group partitioning—participants above and below 25.5 years of age. Older participants completed a significantly smaller fraction of the tests, issuing fewer CFFs (ANOVA; first test $F = 6.512$, d.f. = (1, 103), $p < 0.05$; second test $F = 16.826$, d.f. = (1, 102), $p < 0.001$) and destroying fewer enemies (ANOVA; first test $F = 6.772$, d.f. = (1, 103), $p < 0.05$; second test $F = 10.573$, d.f. = (1, 102), $p < 0.01$). The same pattern held in the number of CFFs completed during practice. On average, the younger participants took 39% more practice shots than the older participants completed in the same amount of time (ANOVA, $F = 8.665$, d.f. = (1, 104), $p < 0.01$). There were no significant differences in test scores and no interactions between age and experimental condition.

Since older participants were more likely to have no performance and to work more slowly overall, producing fewer practice opportunities and IMP interactions in the same time period, they should have been treated differently during practice. For example, it might be best for such users to bias IMP to avoid repeating hints. More remedial hints might be called for. The causes of error observations might be more likely to relate to
simulator usage mistakes rather than domain knowledge gaps. If possible, a different balance of practice time devoted to the various training topics might be called for. Although age was the most significant factor in the present experiment, other demographic information could also have an impact on instruction and therefore should also be included in future learner models (Shute, 1992; Snow & Swanson, 1992).

In order to implement differential treatment based on demographic data in IMP, one of two strategies could be used. First, the pertinent demographic information could be added to the POMDP learner model as an additional, orthogonal cognitive state, just like the affective states in IMP’s current model. Adding a cognitive state would have the benefit that the POMDP would be able to estimate the value of the new state during training. For example, IMP could determine during training whether a trainee was working slower because of traits associated with older participants, whether the particular person was actually older or not. However, the cost of adding an additional state would be a multiplication of the cardinality of the state space, at least a doubling or worse depending on the state to be modeled.

A second alternative would be to construct separate POMDPs for each demographic group. Under this alternative, each participant would be assigned to the appropriate POMDP before the start of training. Separate POMDPs would cause little state-space explosion because trainees would not be allowed to change group assignment during training, but would make the learner model less flexible than the first alternative because the proper group for each trainee would need to be fixed before initializing the POMDP. This could easily be accomplished with a pre-training questionnaire.
7.3.2.3 *Use the full range of available actions*

Finally, review of IMP’s interactions with learners revealed that at no time during the present experiment did the POMDP recommend an intervention that addressed a trainee’s affect rather than a knowledge gap. Furthermore, in the full version of IMP that included an on-screen menu of questions (the QUI), the POMDP always recommended interventions that popped up on screen as interrupting hints, and never recommended the more subtle interventions that merely highlighted questions in the menu.

Interventions addressing affect were hypothesized to be important because negative affect such as boredom or frustration can hurt trainees’ ability to learn (Section 5.2). As discussed in Section 7.2.7.2, the POMDP learner model did estimate that trainees were in these negative states at some points during training, and in some cases the model’s estimates even matched trainees’ reports. However, IMP’s policy never determined that addressing negative affect would return future benefits sufficient to offset the immediate sacrifice of one turn in doing so.

By a similar token, interventions highlighting questions rather than interrupting learners were modeled as being less likely to cause negative affect such as frustration, but also less likely to correct the targeted knowledge gap. Highlighting questions was also predicted to scaffold help-seeking through the question menu and draw trainees’ attention to that question menu, making them more likely to ask IMP for help in general. It would seem that such actions would be beneficial at some times during practice, but IMP’s policy did not agree.
A clue to the reason for both cases of IMP failing to recommend some action types might lie in the affective impact of addressing a knowledge gap. As modeled, presenting any knowledge-directed intervention that did not clear a knowledge gap, including one that addressed a gap that was not present, was likely to move a trainee into a negative affective state. On the other hand, clearing a knowledge gap was likely to move a trainee into a positive affective state. Therefore, IMP may have determined that the best way to manipulate learner affect was simply to clear gaps as long as it judged any were present.

It is difficult to know whether IMP’s unexpected policy decisions were correct or not. IMP’s policy aimed to optimize training based on the model encoded in its POMDP. Therefore, either the POMDP failed to reflect reality in ways that caused IMP to make poor decisions, or the POMDP was largely correct and IMP’s unexpected decisions should be taken seriously. If the latter is the case, perhaps the effect of negative affect on learning, while estimated for each state based on published literature (Section 5.2), is actually not as impactful on overall learning as expected.

A clear lesson that can be learned from the uncertainty about IMP’s recommendations, though, is the need for a tool to predict POMDP learner model responses to hypothetical learners. Such a tool would be related to the probabilistic generative model used for simulated students (Section 5.2), and would ideally be built from real learner interaction histories using a method such as expectation-maximization (Dempster, Laird, & Rubin, 1977). For example, in IMP’s situation, an author could use the tool to tweak the affective state transition model and determine what parameters make
IMP use the action types that it did not use in the real data. If small changes suffice, perhaps they are acceptable to include in an updated model, but if larger changes are required, perhaps some assumptions underlying the affective model are incorrect. Such a tool would tend to decrease model errors before deployment and generally increase model authors’ confidence in POMDPs for application in the real world.

7.3.2.4 Summary of POMDP lessons learned

In conclusion, examining individual experiences of interacting with IMP revealed opportunities for improvement to the POMDP learner model. The model could have benefited from parameter tweaks in differentiating between mistakes and slips, structural changes to include information about learner traits like reported age, and possibly changes to make certain action types more likely. Despite these three changes to increase IMP’s effect on learning, though, the statistics reported in the previous section strongly support the claims that IMP did have a significant and substantive overall effect on learning as it was evaluated. The suggested changes might have increased instructional effectiveness even further.

7.3.3 Inquiry Modeling

On average, participants in the Q condition (full IMP, including the QUI, a menu of questions presented during practice) asked 7.6 questions during practice. Due to the way questions were counted, two-step questions such as questions about using the simulator interface were not included in this number.
Since practice spanned 26 minutes, each participant asked questions at a mean rate of 17.5 times per hour. Study participants asked questions more frequently than reports of learners in a classroom setting who asked 0.1 questions per hour, but less frequently than learners interacting one-on-one with a human tutor, who asked an average of 27 questions per hour. The number of questions asked was also much less than achieved by an earlier ITS that offered learners no interaction except through menus of questions (Graesser et al., 1993). Of course, in the present study questions competed with other interaction channels such as IMP-initiated hints and practicing in the simulator.

In terms of both declarative knowledge and simulator performance, learners’ ability to ask questions led to quantitative learning improvements that were discussed in Section 7.2. The full version of IMP produced outcomes significantly and substantively better than an ablated version with no trainee-initiated questions.

Examining results of an exit survey about IMP reveals the QUI changed participants’ perceptions as well as their performance. Most consistently, learners who could ask questions during training reported significantly increased self-efficacy. Interestingly, learners using the full IMP were also significantly more likely to agree that IMP encouraged them when they needed it. Since IMP actually did not initiate any affective interventions to encourage participants, this result suggests that learners were able to draw encouragement on their own initiative from the QUI, for example by asking questions about their progress in training. On the other hand, participants perceived the ablated IMP as more helpful than IMP with the QUI. One explanation might be that learners using the full IMP felt more forced to take control of IMP’s help interactions.
Statistical analysis did not find significant differences in model accuracy between experimental conditions. Questions learners initiated were hypothesized to be a useful source of new information that could update IMP’s learner model, but transcripts of individual participants revealed implementation issues with using questions to increase model accuracy that are discussed in the next section.

Since the present study demonstrated improved outcomes but not improved model accuracy, it is likely that a common way the QUI improved outcomes was by allowing learners to get the help they needed directly through question asking. A session transcript of a trainee who asked several questions during practice is shown in Table 26.

The participant in Table 26 completed the first practice without using the QUI at all. At the start of the second practice, the participant attacked one target but then interrupted practice to ask a question about differentiating friend targets from foes. The participant then continued the session in the same way, interspersing practice with question-asking in a pattern that is presumably more adaptive than simply practicing without questions or asking too many questions and ignoring practice.

In this example, all of the participant’s questions concerned target recognition. Therefore, the participant’s actions counterbalanced IMP’s tendency in some cases to fail to correct target recognition mistakes. Consequentially, even though IMP did not initiate interventions to tutor this trainee on target recognition, the trainee did not make any target recognition mistakes during testing. During testing the trainee’s target selection performance was about average for participants in the Q condition—meaning better than participants in the other conditions. On the other hand, the trainee’s target engagement
was below average. During practice the trainee did not ask any questions that related to target engagement.
Table 26: A typical transcript from a training session where the participant asked questions. As often happened in the P condition also, IMP did not volunteer hints about differentiating friend and foe targets, but in the Q condition the trainee was able to ask questions during practice and address this knowledge gap.

<table>
<thead>
<tr>
<th>Trainee Action</th>
<th>CFF Errors</th>
<th>IMP Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of Simulator Practice</td>
<td>RF</td>
<td>RF was recommended based on the initial belief, before any observations.</td>
<td></td>
</tr>
<tr>
<td>1. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td>Friend1 refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>2. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td>Friend1 refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>3. Start of Domain Practice</td>
<td>Friend1</td>
<td>The previously recommended Friend1 action was deferred to the first allowed time, the start of the next practice.</td>
<td></td>
</tr>
<tr>
<td>4. CFF</td>
<td>Desc1 Ammo1</td>
<td>Desc1</td>
<td>IMP directly addressed the higher-priority of the two gaps observed.</td>
</tr>
<tr>
<td>5. IFF Question</td>
<td></td>
<td>The trainee asked whether a target was a friend or an enemy.</td>
<td></td>
</tr>
<tr>
<td>6. CFF</td>
<td>Priority1</td>
<td>Foe2</td>
<td>Though the Foe2 gap was not observed, Foe2 can account for seeing Priority1, and clearing the observed gaps depends in the model on clearing Foe2 first.</td>
</tr>
<tr>
<td>7. CFF</td>
<td>Priority1</td>
<td>Priority1</td>
<td>The trainee made a second Priority error and IMP addressed it directly.</td>
</tr>
<tr>
<td>8. CFF</td>
<td>Priority2</td>
<td>Priority2</td>
<td>IMP addressed an observed error.</td>
</tr>
<tr>
<td>9. IFF Question</td>
<td></td>
<td>The trainee asked whether a target was a friend or an enemy.</td>
<td></td>
</tr>
<tr>
<td>10. CFF</td>
<td>Priority1</td>
<td>Desc2</td>
<td>Priority gaps were still observed, but since IMP had addressed them before, it believed they could be slips and instead addressed the Desc2 observation.</td>
</tr>
<tr>
<td>11. IFF Questions (x4)</td>
<td>Priority1</td>
<td>Desc2</td>
<td>The user asked four questions about differentiating friends from enemies.</td>
</tr>
<tr>
<td>12. CFF</td>
<td>Priority2</td>
<td>Desc2</td>
<td>The trainee repeated all the errors IMP had already observed and addressed before. IMP chose to repeat its Desc2 hint.</td>
</tr>
<tr>
<td>13. Shot Miss</td>
<td>Hurry</td>
<td>IMP believed no gaps remained.</td>
<td></td>
</tr>
<tr>
<td>14. CFF</td>
<td>Desc1</td>
<td>IMP repeated a hint to address Desc2, even though Desc1 was observed.</td>
<td></td>
</tr>
<tr>
<td>15. Shot Miss</td>
<td>Hurry</td>
<td>IMP believed no gaps remained.</td>
<td></td>
</tr>
</tbody>
</table>
7.3.4 Inquiry Modeling Lessons Learned

Inquiry modeling in the IMP adaptive trainer produced improved self-efficacy and improved learning by letting trainees ask questions during practice. However, two obstacles kept inquiry modeling from being more effective. First, learners sometimes did not ask enough questions during practice. Second, IMP’s model gave too much weight to learner questions as evidence of learning.

7.3.4.1 Encourage learners to ask questions

Although the mean number of questions asked during practice was 7.6, 11 of the 34 participants in the Q condition did not ask any questions at all. As a result, the median number of questions asked was only 4.5. Anecdotally, several participants even asked an investigator during the experiment how to close or hide the QUI. (They were told that the QUI is always visible during practice.) Participants were not instructed in any way as to using the QUI, so it may be that explicit encouragement before practice would have made the QUI more useful.

Besides those participants who opted not to use the QUI at all, several participants also asked no questions during practice but only at the end of practice. They did not necessarily wait for all targets to be destroyed, but possibly practiced until bored and then turned to the QUI for a change of pace. This pattern of use is not necessarily maladaptive because these trainees concentrated on practice while still asking questions at a time that was organic to their perceived needs during training. However, questions asked after the
end of all attempts to practice could not affect model accuracy or allow IMP to make practice more effective by observing more about the participant.

In general, participants who read more hints performed better (Section 7.2.3).

However, participants chose not to request hints from IMP all too often. More than merely forgetting to ask for help with the QUI, at least some participants actively rejected the opportunity by stating a wish to close it. The experiences of many individuals in using the QUI constituted anecdotal evidence reconfirming the finding that effective help-seeking is a skill learners do not always demonstrate. Additional scaffolding or other support should be added in learning situations when help-seeking is important.

7.3.4.2  Do not rely on help the user initiated

Studying individual experiences of participants who asked many questions during practice suggested that IMP often had highly over-optimistic estimates of their knowledge state. The act of reading a hint acted as evidence that the learner now knew the related material. This evidence was interpreted too strongly in IMP’s learner model. As a result, IMP often presented participants who asked more than a few questions with only advanced hints. In the worst cases IMP did not present useful hints but wasted hint opportunities on repeatedly urging participants to hurry up or keep practicing, behaviors which only make sense when a trainee has no knowledge gaps (Section 7.2.5).

Logs of participants who asked many questions were examined. All four participants who asked 20 or more questions during practice received no useful hints initiated by IMP. Participants who asked between 10 and 20 questions during practice
received all or mostly advanced hints, but some still received basic hints. As a silver lining, some participants who asked the most questions and therefore received no useful help from IMP’s hint actions still performed well on written and simulator tests. The questions they asked may have been sufficient to support their learning on their own initiative, even when IMP did not initiate effective instruction.

An example of trainee questions unbalancing IMP’s learner model appears in the transcript in Table 27. The participant asked only six questions, but IMP misinterpreted them as evidence that the learner did not have knowledge gaps. Five of the questions asked covered extension from the basics taught during training. The trainee asked the questions at the end of the first simulator session, after destroying the enemy targets and presumably while waiting for the practice time to end. IMP interpreted learners asking these questions as evidence they already understood the material taught during training. However, this assumption was unwarranted based on the timing of the questions, and in any event was clearly given too much weight. When the participant later demonstrated basic errors, IMP presented almost entirely hints about advanced topics or hints that were not useful (*Hurry*). IMP also presented the same hint addressing an advanced knowledge gap multiple times, even though the participant never gave any evidence that gap was present.

In addition to anecdotal evidence that questions in general were assigned too much weight in IMP’s model, investigator observations during the experiment suggested that QUI usage can actually be a sign of floundering. Participants who reached an impasse during practice sometimes clicked on every question or clicked on questions
repeatedly. IMP would have misinterpreted such interactions as evidence that the participants were learning from the questions they asked. Overall, it still appears possible to draw knowledge and cognitive state information from these help request patterns, but IMP’s model as designed did not include helpful interpretations of them.

In order to guard against misinterpretation when a learner asks a question without reading the answer, IMP did contain a rule to assess hints as unread if they were open for less than a certain amount of time (Section 6.3). Examining interaction logs suggested that this rule was problematic. No bright line was evident in hint viewing durations, suggesting that some hints were read when they were marked as unread. Given the variation in learners’ natural reading speeds, it might be interesting in the future to reconsider the value of any such rule. The two obvious reasons for learners failing to read hints are accidentally opening a hint that is already known, and attempting to game the help system. Since such reasons were not apparent in IMP’s instructional domain, it might be preferable to simply assume hints are read.

Finally, the information available from the context of learner questions was more complex than expected. An inquiry modeling ITS that fully integrates information from help requests might need to be able to determine whether a person learned from asking a question or not, is interested in or could benefit from more material on the topic or not, and so on. One way to collect more of this relevant information might be to follow up immediately after a learner question with an ITS question designed to differentiate learner states such as specific misconceptions. To reduce distraction the follow-up question could be precisely targeted and quick to complete, or could allow the learner to
skip it. The correct answer could then be displayed immediately, making the reply question fit into the learner model with the same structure as an ITS-initiated hint.

Table 27: An example of trainee questions unbalancing IMP’s learner model. At the end of the first practice the trainee used the QUI to ask all available questions about extension material. Following its learner model, IMP interpreted the advanced questions as evidence that the learner did not need help on any basic material. Later the trainee made some mistakes, but few enough that IMP dismissed them as slips and did not respond effectively to address them.

<table>
<thead>
<tr>
<th>Trainee Action</th>
<th>CFF Errors</th>
<th>IMP Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start of Simulator Practice</td>
<td>RF</td>
<td>RF</td>
<td>RF was recommended based on the initial belief, before any observations.</td>
</tr>
<tr>
<td>1. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Friend1)</td>
<td>Friend1 refers to material that is not introduced until later, so IMP overrode the POMDP’s recommendation.</td>
</tr>
<tr>
<td>2. Steps Question</td>
<td></td>
<td></td>
<td>After destroying one target, the trainee asked what to do next. IMP directed the trainee to attack the other target and the trainee did so.</td>
</tr>
<tr>
<td>3. CFF</td>
<td>No Errors</td>
<td>no-op (ignored Foe1)</td>
<td>The trainee’s question changed IMP’s recommendation from Friend1 to Foe1.</td>
</tr>
<tr>
<td>4. Extra Questions (x5)</td>
<td></td>
<td>Hurry</td>
<td>The trainee read all five of the questions that extend the required material.</td>
</tr>
<tr>
<td>5. Start of Domain Practice</td>
<td></td>
<td>Priority2, Fire1</td>
<td>The fact that the trainee read several Extra questions caused IMP to believe no more basic gaps existed, rendering IMP’s later interactions ineffective.</td>
</tr>
<tr>
<td>6. CFF</td>
<td>Priority2</td>
<td>Ammo1</td>
<td>The trainee made a somewhat basic error, so IMP decided gaps did exist, but addressed a knowledge gap more advanced than either of those observed.</td>
</tr>
<tr>
<td>7. CFF</td>
<td>Priority1</td>
<td>Ammo2</td>
<td>The trainee continued to make another basic error while IMP continued to address advanced gaps instead.</td>
</tr>
<tr>
<td>8. CFF</td>
<td>No Errors</td>
<td>Ammo2</td>
<td>IMP repeated an intervention addressing a gap the trainee never demonstrated.</td>
</tr>
<tr>
<td>9. CFF</td>
<td>Priority1</td>
<td>Foe1</td>
<td>IMP finally started a sequence to address the Priority errors, beginning with Foe1 on which they depend.</td>
</tr>
<tr>
<td>10. Long Idle</td>
<td></td>
<td>Ammo2</td>
<td>IMP skipped the rest of the hints that would address Priority gaps, instead repeating the Ammo2 intervention.</td>
</tr>
<tr>
<td>11. CFF</td>
<td>No Errors</td>
<td>Desc2</td>
<td>IMP again presented an intervention targeting a gap which was probably not present.</td>
</tr>
<tr>
<td>12. Shot Miss</td>
<td></td>
<td>Ammo2</td>
<td>IMP repeated Ammo2 a fourth time.</td>
</tr>
</tbody>
</table>
7.3.5 Discussion Summary

This section related anecdotal evidence to build an impression of how IMP might have achieved the results presented in Section 7.2. A transcript of a typical session included examples of IMP responding to a learner’s needs and planning ahead to make interventions more effective. Another transcript showed how a learner could take control of instruction by asking questions with the QUI. However, there were several ways in which the individual experiences could have been improved. Therefore this section also described several lessons learned that could bring future improvement for IMP’s POMDP learner model and inquiry modeling interactions.
CHAPTER 8: CONCLUSION

This dissertation describes original research that led to improved understanding of intelligent tutoring systems (ITSs). Compared to non-adaptive instruction tools, ITSs are effective in teaching learners, but they are not as effective as human tutors. Furthermore, the most effective ITSs are costly to develop. The present dissertation improves the state of the art in the science and application of ITSs.

This dissertation introduces inquiry modeling, a new channel for ITSs to receive user model inputs from users’ help requests. User questions and other requests for help are an important part of instruction for human tutors that is underused in existing ITSs. The dissertation explores the kinds of questions users can be expected to ask, what those help requests can tell an ITS about each user, and what other points ITS designers should consider when adding freer user questions and inquiry modeling abilities to an ITS.

This dissertation also describes novel work in using partially observable Markov decision processes (POMDPs) as ITS user models. POMDPs are a powerful class of representations that allow planning under uncertainty. Since planning ahead can improve instruction and tutoring in general involves many sources of uncertainty, POMDPs are well suited to controlling ITSs. However, POMDPs are prone to important scaling issues that require specialized representations in order to model the large problems that ITSs typically face.

The following sections summarize the contributions of the dissertation research.
8.1 Summary of the Literature Review and Experts’ Questionnaire

A review of the archived literature synthesizes knowledge in the current state of the art about intelligent tutoring systems and their user models. An anonymous survey of ITS practitioners and researchers adds to knowledge about development efficiency and other practical considerations that may be less widely discussed in the existing literature.

POMDPs as learner models are compared to more widely used model types. POMDPs are more abstract and therefore easier to develop than detailed models. On the other hand, POMDPs’ ability to plan despite uncertainty gives them the potential to produce a larger effect on learner outcomes than simple models.

8.2 Summary of Support for the Research Hypotheses

The work supports four general research hypotheses: that letting users ask questions will improve learning, that tractable POMDPs can be designed to represent the tutoring process, that planning interaction sequences will improve pedagogical interventions, and that these changes together will improve instructional efficacy.

The two studies with human participants showed inquiry modeling can improve learning outcomes. Learner questions provided a new source of information about the obstacles to learning and opportunities for imparting additional material. In the full-scale human study, letting learners ask questions improved their self-efficacy and their performance outcomes. However, the learner questions did not improve the ITS’s model accuracy in this case because of implementation issues. This finding demonstrates that applying inquiry modeling to model improvement is more complex than envisioned.
Several studies with simulated learners demonstrated that with suitable ITS-specific representations, POMDPs can scale to function effectively on problems as large as real-world ITSs face. The new representations were insensitive to a wide range of parameter values and could potentially be applied to many different tutoring problems. The creation of a POMDP ITS to train a realistic military task provided a fully implemented example of a POMDP scaling up to interact effectively with learners in a realistic ITS domain.

Evaluating the new POMDP ITS provided evidence in support of the planning hypothesis and summative hypothesis. Rather than following a simple policy such as tutoring the last error observed, the ITS demonstrated responsive and adaptive interaction patterns. Training with the ITS improved learner outcomes on a variety of measures. Learners achieved better performance, improved declarative knowledge, and reported more positive feelings about the trainer and their own training readiness.

8.3 Dissertation Summary

Intelligent tutoring systems are powerful tools for instruction. Teachers and trainers in current practice must often work in classrooms, lecture halls, and other large group environments that never allow for the individual attention learners need to reach their full potential. By modeling individual learners and determining how best to help each one, intelligent tutoring systems can bring adaptive instruction to many more students and trainees.
This dissertation advances the science and technology of intelligent tutoring. The studies it includes bring new understanding of how planning under uncertainty and interacting with user questions can improve intelligent tutors. The research directly led to a tractably scaling application of the new ideas to a real-world problem. These contributions are novel and useful to ITS developers. Together, this work helps bring the future of effective and efficient intelligent tutors one step closer.
APPENDIX A: SURVEY OF ITS PROFESSIONALS
This appendix reproduces the text of the questionnaire presented to ITS professionals.

The questionnaire was originally presented as a series of web pages. The appendix depicts the different possible answer types according to the following key:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>The respondent may select at most one of the listed choices to answer this question.</td>
</tr>
<tr>
<td></td>
<td>The respondent may select any number of the listed choices to answer this question.</td>
</tr>
<tr>
<td></td>
<td>The respondent may enter any text to answer this question.</td>
</tr>
<tr>
<td></td>
<td>The respondent may enter any text to answer this question.</td>
</tr>
<tr>
<td></td>
<td>The respondent must choose at most one item from the following list (presented in the order given).</td>
</tr>
<tr>
<td></td>
<td>• Did not use student modeling</td>
</tr>
<tr>
<td></td>
<td>• Overlay model</td>
</tr>
<tr>
<td></td>
<td>• Differential model</td>
</tr>
<tr>
<td></td>
<td>• Perturbation model</td>
</tr>
<tr>
<td></td>
<td>• Bug or bug-part library</td>
</tr>
<tr>
<td></td>
<td>• Model tracing</td>
</tr>
<tr>
<td></td>
<td>• Knowledge tracing</td>
</tr>
<tr>
<td></td>
<td>• Example tracing</td>
</tr>
<tr>
<td></td>
<td>• Other production-rule model</td>
</tr>
<tr>
<td></td>
<td>• Constraint-based model</td>
</tr>
<tr>
<td></td>
<td>• Case-based model</td>
</tr>
<tr>
<td></td>
<td>• Finite-state automata</td>
</tr>
<tr>
<td></td>
<td>• Behavior transition networks</td>
</tr>
<tr>
<td></td>
<td>• Decision trees</td>
</tr>
<tr>
<td></td>
<td>• Neural networks</td>
</tr>
<tr>
<td></td>
<td>• Neurule system</td>
</tr>
<tr>
<td></td>
<td>• Bayesian networks</td>
</tr>
<tr>
<td></td>
<td>• Other (fill in below)</td>
</tr>
</tbody>
</table>
A recent ITS

Please describe the intelligent tutoring system (ITS) you worked on most recently that is ready, or nearly ready, to interact with students.

1. For the ITS you worked on most recently, approximately how many different student models did it use?
   - [ ] No explicit student model
   - [ ] 1
   - [ ] 2
   - [ ] 3 or more modeling components

2. What student model type or modeling algorithm did the system use to SELECT MATERIAL to present? What did the system use to RESPOND TO ERRORS? If the system used more than one student model, please describe ONE model for each adaptation type.

   Selecting or ordering material:  
   Adapting corrections or hints:  
   
   PRIMARY model or algorithm:  

   Model types or algorithms not on the list:

For the following questions, feel free to answer with an estimate, a range, or even an order of magnitude.

Please measure work in person-hours: each person working full-time for one week contributes about 40 person-hours, and one person working full-time for a year contributes about 2000 person-hours.

3. About how much work, measured in person-hours, did it take to create the ITS? How much of that time was spent working on the student models?

   The whole ITS
   The primary student model for MATERIAL SELECTION
   The primary student model for HINTS AND FEEDBACK
4. Approximately how much additional time, measured in person-hours, was saved by reusing work from other projects?

   The whole ITS
   The primary student model for MATERIAL SELECTION
   The primary student model for HINTS AND FEEDBACK

5. Did your team use any authoring tools to help build the ITS?

   Yes          No
   The whole ITS
   The primary student model for MATERIAL SELECTION
   The primary student model for HINTS AND FEEDBACK

6. (Optional) If so, which authoring tools did you use?

7. When the project was finished, how many hours of instruction per student did the ITS provide?

8. Are there any other comments you’d like to include about the student models in this ITS, how their design was determined, the model-building process, or anything else?
**Overlay models**

9. Have you ever worked on an ITS that used an OVERLAY MODEL?
   - [ ] Yes
   - [x] No

10. How satisfied were you with the OVERLAY MODEL's performance?
    - Very satisfied
    - Very dissatisfied
    - N/A

  Performance:  ⬜  ⬜  ⬜  ⬜  ⬜  ⬜  ⬜  ⬜

11. Approximately how many person-hours did the OVERLAY MODEL take to construct?

    

12. (Optional) What was the name of the ITS that used an OVERLAY MODEL?

    

Buggy models

13. Have you ever worked on an ITS that used a DIFFERENTIAL MODEL, a PERTURBATION MODEL, a BUGGY LIBRARY, or a BUG-PART LIBRARY? For convenience, we will group these different model types together under the general heading of BUGGY MODELS.

☐ Yes  ☐ No

14. How satisfied were you with the BUGGY MODEL's performance?

Performance:

☐ Very satisfied  ☐ Very dissatisfied  ☐ N/A

15. Approximately how many person-hours did the BUGGY MODEL take to construct?

16. (Optional) What was the name of the ITS that used a BUGGY MODEL?
Production rules and model tracing

17. Have you ever worked on an ITS that used a COGNITIVE PRODUCTION-RULE model, or another model that performed COGNITIVE MODEL-TRACING?
☐ Yes ☐ No

18. How satisfied were you with the PRODUCTION-RULE MODEL's performance?
Performance: ☐ Very satisfied ☐ Very dissatisfied ☐ N/A

19. Approximately how many person-hours did the PRODUCTION-RULE MODEL take to construct?

20. (Optional) What was the name of the ITS that used a PRODUCTION-RULE MODEL?


**Knowledge tracing**

21. Have you ever worked on an ITS that used a model that performed KNOWLEDGE-TRACING?
   - Yes
   - No

22. How satisfied were you with the KNOWLEDGE-TRACING model's performance?
   - Very satisfied
   - Very dissatisfied
   - N/A

   Performance:

23. Approximately how many person-hours did the KNOWLEDGE-TRACING model take to construct?

24. (Optional) What was the name of the ITS that used a KNOWLEDGE-TRACING model?

**Constraint-based models**

25. Have you ever worked on an ITS that used a CONSTRAINT-BASED model?
   - [ ] Yes
   - [ ] No

26. How satisfied were you with the CONSTRAINT-BASED model's performance?
   - [ ] Very satisfied
   - [ ] Very dissatisfied
   - [ ] N/A

   **Performance:**
   [ ] [ ] [ ] [ ] [ ] [ ] [ ]

27. Approximately how many person-hours did the CONSTRAINT-BASED model take to construct?

28. (Optional) What was the name of the ITS that used a CONSTRAINT-BASED model?


Bayesian networks

29. Have you ever worked on an ITS that used a BAYESIAN NETWORK model?

☐ Yes  ☐ No

30. How satisfied were you with the BAYESIAN NETWORK model's performance?

Very satisfied  Very dissatisfied  N/A

Performance:  ☐  ☐  ☐  ☐  ☐  ☐  ☐  ☐

31. Approximately how many person-hours did the BAYESIAN NETWORK model take to construct?

32. (Optional) What was the name of the ITS that used a BAYESIAN NETWORK model?

________________________________________

237
Classifiers as student models

33. Have you ever worked on an ITS that used a decision tree, neural network, finite state machine, or other CLASSIFIER as a student model? If so, what kinds of classifiers did this ITS use?

34. Did the CLASSIFIER use any machine learning algorithms to modify the model, either before or after deploying the ITS?

☐ Yes, before any students ☐ Yes, while students used ☐ No, did not use machine learning
used the ITS the ITS

35. How satisfied were you with the CLASSIFIER's performance?

Performance: ☐ Very satisfied ☐ ☐ ☐ ☐ ☐ Very dissatisfied ☐ N/A

36. Approximately how many person-hours did the CLASSIFIER take to construct?

37. (Optional) What was the name of the ITS that used a CLASSIFIER as a student model?
Demographic information

As with all the questions in this survey, these questions are optional and you may leave any of them blank.

38. What type of organization do you work for?
☐ Industry ☐ Government ☐ Academic

39. Approximately how many adaptive education or training systems have you been involved with researching or creating?
☐ 0 ☐ 1-2 ☐ 3-5 ☐ 6+

40. Approximately how long have you been involved with the research or development of adaptive technologies for education or training?
☐ N/A ☐ 1-2 years ☐ 3-6 years ☐ 7+ years

41. Finally, are there any other comments you would like to add about any aspect of creating student models for intelligent tutoring systems?
APPENDIX B: IRB APPROVAL FOR HUMAN STUDY 1
Approval of Human Research

From: UCF Institutional Review Board #1  
FWA00000351, IRB00001138

To: Sarah L. Schatz

Date: October 08, 2010

Dear Researcher:

On 10/8/2010, the IRB approved the following modifications/human participant research until 08/04/2011 inclusive:

- **Type of Review:** IRB Addendum and Modification Request Form
- **Modification Type:** Cynthia Cakes added to the study as research associate
- **Project Title:** NEW-IT Inquiry Modeling Study I (JT)
- **Investigator:** Sarah L. Schatz
- **IRB Number:** SBE-10-66987
- **Funding Agency:** Office of Naval Research
- **Grant Title:** N00014-08-C-0166, Next-generation Expeditionary Warfare Intelligent Training (NEW-IT)
- **Research ID:** N/A

The Continuing Review Application must be submitted 30 days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form cannot be used to extend the approval period of a study. All forms may be completed and submitted online at https://irb.ucf.edu.

If continuing review approval is not granted before the expiration date of 08/04/2011, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in IRB5 so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Joseph Bielitzki, DVM, UCF IRB Chair, this letter is signed by:

Signature applied by Joanne Muratori on 10/08/2010 02:20:19 PM EDT

IRB Coordinator
APPENDIX C: IRB APPROVAL FOR HUMAN STUDY 2
Approval of Human Research

From: UCF Institutional Review Board #1
FWA0000351, IRB0000138

To: Sae L. Schatz

Date: May 16, 2011

Dear Researcher:

On 5/16/2011, the IRB approved the following human participant research until 5/15/2012 inclusive:

Type of Review: Submission Response for UCF Initial Review Submission Form
Expedited Review Category # 7
This approval includes a Waiver of Written Documentation of Consent

Project Title: NEW-IT Inquiry Modeling Study 2 (JT)
Investigator: Sae L. Schatz

IRB Number: SBE-11-07675

Funding Agency: Office of Naval Research
Grant Title: N00014-08-C-0186, Next-generation Expeditionary Warfare
Intelligent Training (NEW-IT)
Research ID: 1046136

The Continuing Review Application must be submitted 36days prior to the expiration date for studies that were previously expedited, and 60 days prior to the expiration date for research that was previously reviewed at a convened meeting. Do not make changes to the study (i.e., protocol, methodology, consent form, personnel, site, etc.) before obtaining IRB approval. A Modification Form cannot be used to extend the approval period of a study. All forms may be completed and submitted online at https://iris.research.ucf.edu.

If continuing review approval is not granted before the expiration date of 5/15/2012, approval of this research expires on that date. When you have completed your research, please submit a Study Closure request in IRIS so that IRB records will be accurate.

Use of the approved, stamped consent document(s) is required. The new form supersedes all previous versions, which are now invalid for further use. Only approved investigators (or other approved key study personnel) may solicit consent for research participation. Participants or their representatives must receive a copy of the consent form(s).

In the conduct of this research, you are responsible to follow the requirements of the Investigator Manual.

On behalf of Kendra Dimond Campbell, MA, JD, UCF IRB Interim Chair, this letter is signed by:

Signature applied by Joanne Maratori on 05/16/2011 03:40:35 PM EDT

Page 1 of 2
Participants answered each survey question on a seven-point Likert scale as follows:

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>🅿</td>
<td>🅿</td>
</tr>
<tr>
<td>🅿</td>
<td>🅿</td>
</tr>
<tr>
<td>🅿</td>
<td>🅿</td>
</tr>
<tr>
<td>🅿</td>
<td>🅿</td>
</tr>
<tr>
<td>🅿</td>
<td>🅿</td>
</tr>
<tr>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

1. I can remember the knowledge and skills from the teaching videos and the practices.
2. I feel ready to apply the knowledge and skills from the videos and practices in new situations.
3. The teaching videos explained everything I needed to know to do a good job.
4. The material was easy.
5. It was easy to learn from the teaching videos.
6. It was easy to learn from the practice trainer.
7. I tried hard to learn as much as I could during practice.
8. I tried hard to do my best on the tests.
9. The practice trainer worked the way I expected it to.
10. The practice trainer hurt my ability to learn or perform well.
11. The practice trainer understood what help I needed.
12. The practice trainer gave me useful hints or reminders.
13. The practice trainer told me things I already knew.
14. The practice trainer didn’t help me at times when I needed it.
15. During practice, the trainer distracted me from working or remembering.
16. The practice trainer encouraged me when I needed it.
17. I would like to learn more material similar to what I practiced today.
18. I would like to learn more from this particular practice trainer.
19. I could control the practice trainer.

20. I had enough time to practice before the tests.

21. I could have learned or performed better if I could ask more questions during practice.

22. I was interested in the material.

23. I got high scores on the tests.

24. I felt satisfied with my performance on the tests.
LIST OF REFERENCES


International Conference on User Modeling, Adaptation, and Personalization, Big Island, HI.


Murphy, K. (2002). Representing hierarchical POMDPs as DBNs, with applications to mobile robot navigation. Boston, MA: Massachusetts Institute of Technology.


