Evaluating Software Agents Using Human Benchmarks

Robert D. Grant, Dewayne E. Perry
The University of Texas at Austin, Austin, TX, USA
{bgrant,perry}@mail.utexas.edu

***Abstract*—**

Background: **Software agents are becoming increasingly common in the engineering of software systems. In this work, we explore using human subjects to create benchmarks for evaluating these agents. In our case studies, we address the domain of instructable software agents as proposed by the Bootstrapped Learning project [1].**

Aim: **Our aim is to define and refine requirements, problem solving strategies, and evaluation methodologies for software agents, paving the way for rigorous experiments comparing their performance with human benchmarks.**

Method: **Little was known about what factors would be critical, so our empirical approach is exploratory case studies. In two studies covering three distinct groups, we use human subjects to develop an evaluation curriculum for instructable software agents, collecting quantitative data through online quizzes and tests and qualitative data through observation.**

Results: **Though we provide some analysis of quantitative data, our most important results are qualitative. We uncover and address several intrinsic challenges in comparing software agents with humans, including the greater semantic understanding of humans, the eidetic memory of software agents, and the importance of various study parameters (including timing issues and lesson complexity) to human performance.**

Conclusions: **This work provides valuable insight into evaluating software agents with human benchmarks. We hope future researchers will be able to perform controlled experiments in various domains using a methodology based on the results of our case studies.**

I. Introduction

Software agents are becoming increasingly common in the engineering of software systems. These agents are generally intended to be autonomous and independent, and may be specialized to a particular domain or required to function in unforeseen domains. There are a variety of techniques for creating that autonomy; in this paper we target domain-independent human-instructable agents. Whatever the approach used, we should rigorously evaluate agent performance through empirical studies. In this research we explore the use of humans in creating benchmarks for the evaluation of software agents.

II. Study Aims

In this paper we present two exploratory case studies with human subjects wherein we define and refine requirements, problem solving strategies, and evaluation methodologies for instructable software agents (*e-students*) as proposed by the Bootstrapped Learning project [1]. The eventual goal is to directly compare agents performance with human performance on "identical" curricula in controlled experiments. This paper is a more complete version of preliminary publications [2,3] in which we could not yet reveal details of the hidden domain of instruction. We refer to these studies as "Phase II" and "Phase III" with respect to a preliminary "Phase I" study conducted earlier by our group and not covered in this paper [3,4].

III. Related Work

Since this work concerns evaluating instructable agents rather than developing them, there is little previous work that is directly applicable. In this section, we begin with an overview of the Bootstrapped Learning project and our work's relation to it, and we then give a short overview of other related work.

A. The Bootstrapped Learning Project

Bootstrapped Learning (BL) is a program originated by DARPA exploring a new direction for machine learning in which human instructors teach, rather than program, electronic computational agents [1,5].

Two teams are working in parallel as part of this project. Our group is part of the Curriculum Team, which also includes groups from BAE Systems [2,3], Stottler Henke Associates Inc. [6,7], Cycorp Inc., and others. The Curriculum Team is developing the Bootstrapped Learning Analysis and curriculum Development Environment (BLADE), which includes developing a framework to support BL, a set of laddered curricula across a variety of domains as testing vehicles for the e-student, and an evaluation of the e-student on both hidden and known domains. Other groups, as part of the Learning Team, are developing the learning agents [8]–[10].

BLADE includes three agents, whose interactions and relationships are shown in Fig. 1. A teacher agent serves as a proxy for an eventual human teacher, instructing and testing the e-student. The student agent is the embodiment of the e-student, which typically employs a number of learning algorithms. The world agent serves as a proxy for a domain simulator. Over the first three phases of the BL program, the Curriculum Team has developed sets of curricula in a variety of complex domains including planning robotic arm movements in a Blocksworld [3,4], robot soccer [11,12], unmanned aerial vehicle surveillance, diagnosis tasks for the international space station [6], armored task force maneuvers, and diagnosing and repairing problems with a satellite-tracking ground station (the domain explored in this work).

BLADE employs InterLingua (IL) and InteracTion Language (ITL), developed specifically for the BL project [1], to

pass messages between agents in the BLADE framework. It uses an automated teacher, rather than a human teacher, to ease testing scalability and reproducibility. Part of the Curriculum Team's research is to explore how best to incorporate a human teacher.

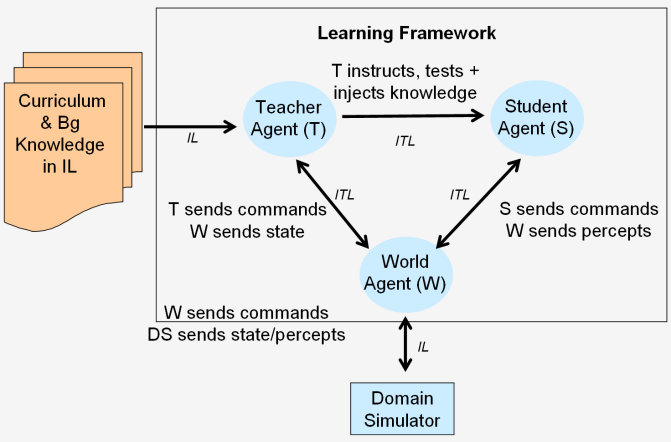

Fig. 1. The BLADE Framework

B. Computer Tutoring

The area of computer tutoring can be seen as an inverse problem to what we are investigating. In particular, the area of teachable agents bears some surface similarity to BL. In this field, human students teach learning agents in order to improve their own understanding of concepts (*Learning by Teaching*). One example is the *Betty's Brain* system [13]. However, in these systems the importance is placed on how well the human instructor learns, not on the capabilities of the learning agent.

C. Human Learning, Teaching, and HCI

Our group previously performed a case study to determine how a human would attempt to teach an e-student [3,4]. We leveraged the results of this study to create the human benchmark studies outlined in this paper.

We leverage existing knowledge about human learning and computer-human interaction when giving recommendations for future benchmarking experiments [14]–[16].

IV. Phase II: Initial Human Comparison Study

In Phase II of this project, the overall plan was

1) to establish a laboratory setup in which to create benchmark tests using human students and a human-consumable representation of the e-student instruction, and
2) to create the necessary experimental protocols by which to create those benchmark tests for the e-student.

Our overarching goal was to mimic the e-student context as closely as possible in the human student studies so an e-student can be fairly but rigorously evaluated with respect to its ability to learn from the defined curricula.

As we knew very little about the curriculum complexity for which human students could achieve a satisfactory level of learning, this study was an exploratory case study. Our use of human students in this study was solely to establish the benchmark lessons and tests. We were not studying the human students at all; we were only using them to establish the levels of lessons and tests to which the e-students shall be held in evaluating their performance.

A. Study Objectives

Our goal was for this case study to provide us with a set of benchmark lessons and tests by which to measure an e-student's ability to learn from an automated teacher. We sought to produce lessons and tests on which human students could achieve a minimum score of 80%. To ensure the internal validity of the study, it was imperative that the lessons and tests given to the human students matched the corresponding units for the e-students as closely as possible, through differences between the student types made this somewhat difficult.

B. The Hidden Domain

The *hidden domain* for this study was diagnosing and fixing problems with a satellite-tracking ground station. In the real-world counterpart to our hidden domain, satellites in orbit are monitored and controlled from mission-control centers on the ground. These mission-control centers communicate with satellites through ground stations that operate antennas. The training scenarios in our study's hidden domain consisted of diagnosing and repairing misconfigurations and component failures in ground station equipment. These scenarios and the simulation interface are drawn from actual experiences with ground stations for scientific satellites.

Students interacted with the domain through a Java simulator with a Graphical User Interface (GUI), as shown in Fig. 2. Each lesson was available in up to three different instructional styles, known as Natural Instruction Methods (NIMs). Respectively, these methods presented material

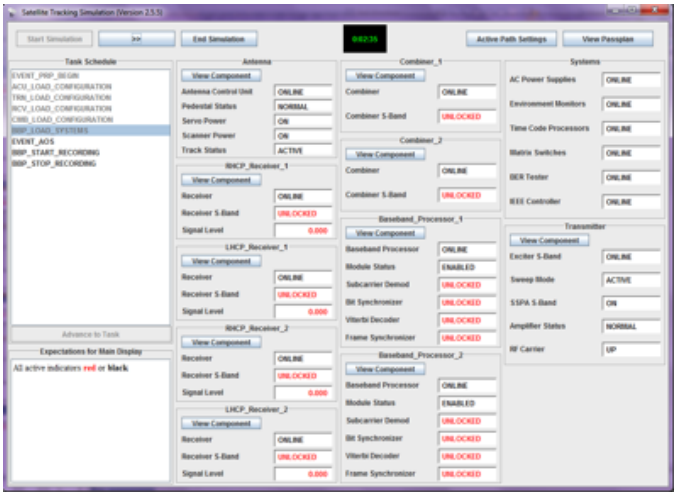

Fig. 2. Hidden Domain Simulator

1) *by telling* - by giving the definition of a task to be done
2) *by example* - by showing screenshots of tasks being done in the simulator, and
3) *by feedback* - by instructing students to perform a task and then providing feedback about whether the student performed the task correctly.

In this phase, we gave each human student all three lesson types, with the option to skip any particular lesson type on any curriculum unit or rung.

C. Initial Study Design

Our first design was a direct analog of the relationship between an automated teacher and an e-student. Since we were concerned with evaluating the curriculum and not the subjects, a major concern was preventing a human teacher from unconsciously providing extra-curricular information to the student through facial expressions, gestures, tone, etc. In this design there was one teacher, one student, and at least one observer per session. The teacher and the student were each provided with two Windows Vista laptops and one external monitor. These setups were positioned on desks facing one another but separated by a screen (as shown in Fig. 3). One laptop on each side was used for instant messaging between teacher and student, and the other was used for manipulation of the hidden domain simulator. The external monitor on each side was connected to the hidden domain simulator laptop on the other side of the separating screen. This allowed a student to see the teacher's example usage of the simulator and also allowed the teacher to observe a student's simulator practice and tests.

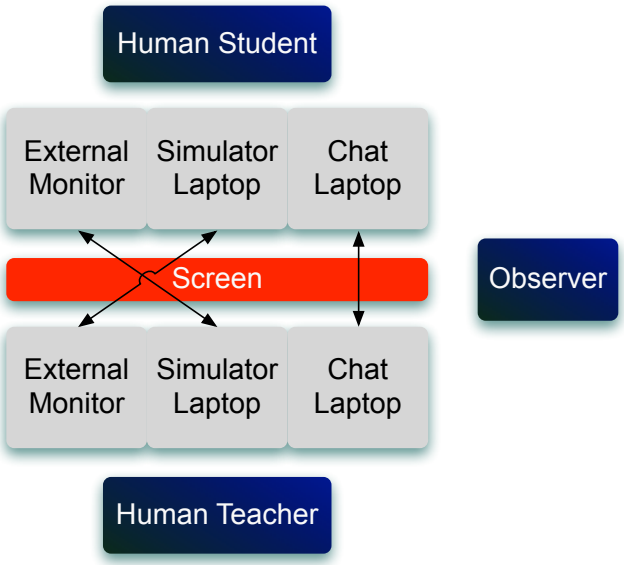

Fig. 3. Phase II: Initial Testing Setup

The student and teacher were only allowed to communicate through the electronic means provided. The student was allowed to talk to the observer about practical issues like the need for breaks, equipment or software failures, or desire to withdraw from the study. For teaching, the human teacher simply typed transliterations of the electronic curriculum into the chat window and performed curriculum examples in the simulator on the corresponding laptop. We used a screen-drawing tool to allow the teacher to emphasize certain areas of the screen by circling and underlining. The student was only allowed to ask the teacher to skip or repeat a lesson; no other communication was allowed, as there is none between the e-students and automated teachers. After testing this setup with a few volunteers, we uncovered significant problems.

D. Problems with Initial Design

There were several practical problems with this initial setup and instantiation of the curriculum. First, because of the restricted communication method, there was no way for the teacher to tell if a student had seen and understood an utterance or demonstration, so the teacher could not pace the curriculum correctly. Also, the teacher's job of typing utterances and giving demonstrations was time-consuming, extremely error-prone, and initially there was no protocol that allowed the teacher to report errors or redo instructions.

These minor issues were easy to fix, but there were deeper problems with teaching human students in this way (with a direct translation of the electronic curriculum). Mainly, this method of instruction was very slow, and we were overrunning our allotted study time of four hours per person by a large margin. Also, both the method of communication and the phrasing of instructions were sometimes awkward, frustrating, and misleading for human students.

To solve these problems, we thought carefully about the differences between electronic and human students and considered how we could improve the human testing process while ensuring that the curriculum we were evaluating was still equivalent to that given to the e-student.

One major issue was that e-students have the advantage of eidetic memory. This means that an e-student is able to recall any given lesson exactly, whereas a human student is not. On the other hand, human students have a much deeper understanding of language. This knowledge can sometimes lead a human student to glean more from a given instruction than an e-student could, but it can also confuse a human student when an instruction may have multiple meanings, or when an instruction is unnaturally phrased (though strictly correct).

E. Final Study Design

Our main modification that rendered human testing feasible was to make the curriculum self-paced. Instead of a human teacher feeding every line of curriculum to the student and demonstrating simulator usage manually, we formatted the curriculum as PowerPoint slides with instructions and accompanying figures. For the feedback/practice lessons, we provided the student with instructions on procedures to try in the simulator and what the outcome should look like if the

procedure was performed correctly; we called this the *choose-your-own-adventure* style.

This eliminated several complexities in our initial laboratory design. Instead of two laptops and an external monitor for both student and teacher, a student could now go through the curriculum on a single laptop, and a teacher/observer could watch either over a subject's shoulder or electronically through Microsoft Remote Desktop. We no longer needed the awkward and restricted text communications channel, and we no longer needed to worry about issues of pacing or about instructor error. This format had the additional advantage of allowing us to run multiple students in parallel with only one teacher/observer.

We addressed the issue of the electronic students' eidetic memory by allowing human students to review previously viewed lessons at any time, and by allowing students to take notes, either by hand (on scratch paper) or in Notepad on their laptop.

The issue of awkward and/or ambiguous language was harder to solve, but we overcame it by beta testing our curriculum with several students and by changing the offending parts based on feedback. We ended up using a much more "natural" expression of the instructions for human testing despite the small risk of deviating from the electronic curriculum, because the direct translation from the e-curriculum was simply too confusing for human students. Moreover, where we had encountered semantic laden terms (such as the names of lessons), we substituted neutral language—for example "lesson blue" instead "fix the X lesson".

While self-pacing did decrease the overall time for a student significantly, there was no escaping the fact that the curriculum was simply too long for a human student to complete in four hours. In our final version, human students learned and were tested on a much-reduced, but still representative, sampling of the curriculum than the e-students must endure.

F. Participant Selection and Scheduling

The study participants were volunteer respondents to mass emails sent to the graduate student mailing lists at The University of Texas at Austin (UT). The participants were selected on a first-come first-served basis, and those who qualified were offered $75 if they completed the study or $7.50 per hour if they withdrew or were unable to finish. Participants were told the study would take approximately three hours, and that they would have a maximum of four hours to complete the study. They were told they might be asked to leave if they exceeded four hours or failed a quiz more than twice.

A maximum of two subjects were tested concurrently, and we held at most two testing sessions per day: one at 9:00 AM and one at 1:00 PM, for a total of four subjects per day. Participants were assigned testing times based on their preference and availability.

G. Benchmark Procedure

Before students came to the lab they were provided with basic instructions and a broad overview of how the study would proceed. The schedule for a participant is listed in Table I; times are approximate.

TABLE I
PHASE II SUBJECT SCHEDULE

Minutes	Material
15	Introduction and background
15	Pre-test
180	Lessons and Quizzes
30	Post-test

When subjects first arrived, they were required to fill out the requisite forms with their payment information. Then, an instructor gave a short presentation on necessary background knowledge. This was knowledge that human subjects needed to know before beginning the true curriculum and mostly consisted of an overview of the hidden domain simulator GUI. This was not part of the evaluated curriculum; the e-student has no similar interface to the hidden domain. No curriculum material was covered by this point.

Then, the observer seated each student at a workstation and administered a short pre-test. In this test, the observer opened the hidden domain simulator for each student (remotely, using Microsoft Remote Desktop), chose a single test scenario (out of five possible scenarios), and each student was given five minutes to complete the test. The true test-scenario labels were replaced with numbers in the student-facing version, but nonetheless, we required the students to face away from their monitors while the observer selected a scenario. The study preference was for the human student to get no more than 20% on the pre-test, thereby assuring us that the results on the post-test were the results of the lessons and not prior knowledge.

Next, the students were allowed to begin the self-paced curriculum. The student proceeded by opening and reading curriculum files in the order indicated on their provided Student Instructions form. This curriculum consisted of PowerPoint slides (the lessons), interspersed with five multiple-choice web-browser-based quizzes that tested students' understanding of previous units. A student was not allowed to continue on to further lessons until he or she scored a minimum of 80%, and if a student should fail the same quiz more than twice, the subject would be asked to leave the study.

The observer was able to watch the students from his own workstation and view their screens remotely using the Microsoft Remote Desktop software. The observer was not allowed to answer any questions about the material itself, so students were forced to rely on the curriculum for all of their domain knowledge.

Finally, after students successfully completed all the lessons and quizzes, they were given a post-test. The post-test was similar to the pre-test, except that the student was given five different test scenarios out of six possible (instead of a single scenario selected out of five possible). The sixth scenario was a *Null* scenario we deemed unsuitable for a pre-test. The scenarios were administered in a random order, computed

with the Python `random.sample` function before subjects arrived. As in the pre-test, a student only had five minutes to successfully complete each test. At the observer's discretion, students were then given a more difficult test.

To avoid associating personal information about the study participants with their study data, the observer assigned each subject a monotonically increasing Subject ID number and recorded all subject data under this identifier. To help the observer, a Subject Data form was prepared in advance for each subject with random values for a session pre-computed (such as pre-test scenario number and post-test scenario numbers and order). This form also had blanks for recording all relevant data for a subject, including:

- Testing Date
- Testing Group (History or Electrical & Computer Engineering)
- Start Time
- End Time
- Pre-test scores
- Quiz scores
- Post-test scores
- Notes

H. Quantitative Results

A total of 5 students were used in preliminary dry runs, 4 of which were Graduate Research Assistants under Professor Perry, one of which was a Post-Doctoral Student under Professor Perry. 31 students participated under the final study design: 6 students were respondents to the History department mailing (Group H), and 25 were respondents to the Electrical & Computer Engineering department mailing (Group E).

Background data about subjects (such as major) was not actually recorded, but most of the H Group subjects were graduate students in History, and most of the E Group subjects were graduate students in Electrical & Computer Engineering with a few students of Computer Science and other engineering disciplines mixed in. The H Group was evenly split on gender with 3 females and 3 males, while the E Group had 22 males and 3 females.

All subjects except for one failed the pre-test; the one who passed confessed to having previous knowledge of the domain. If that student is included, the mean pre-test score is 3%; otherwise, it is 0%.

The mean quiz scores are as follows (where there two scores, the second is the mean of retakes after failing a first try):

The mean score on the post-test scenarios was 92% if the knowledgeable student is kept in; otherwise, the mean score is 91%. 1 to 3 (mean 2) students failed any given post-test scenario.

The time taken for an individual to complete the study ranged from 1:40 to 3:19, with a mean completion time of 2:30.

I. Qualitative Results

The hidden domain curriculum was neither too hard nor too easy. Before going through the curriculum, all students without domain knowledge failed the pre-test. After going through the curriculum, 28 out of 31 students passed the post-test (80% was considered passing).

TABLE II
PHASE II MEAN QUIZ SCORES

Quiz	Mean Score (%)		N	
	Try 1	Try 2	Try 1	Try 2
Q1_1.1	89	98	28	9
Q2_3.1	89	99	28	5
Q2_6.1	98	99	28	1
Q3.4_3.1	99	n/a	28	0
Q3.4_6.1	100	n/a	28	0

Subjects clearly improved their comprehension as they progressed through the lessons. Quiz scores steadily improved, and no students needed to retake quizzes 4 or 5. The student with the lowest post-test score (20%) had problems with the English language, though this was not clear until later in the study when he repeatedly and blatantly ignored instructions. This student was also the only one who came close to being asked to leave the study because of failing quizzes.

The History graduate students did as well as the Electrical Engineering graduate students and took less time on average to complete the study. Thus, it is unlikely that technical bias is a confounding variable. A couple of the History students had more difficulty with the basics of computer operation and required more explanation in the Background Knowledge portion of the study, but this did not seem to affect their post-test scores.

Switching to a self-paced approach significantly reduced the study completion time for the subjects (by a factor of 3 or 4), and the subjects seemed to enjoy the self-paced version much more than the initial version of the study. Several subjects even commented that the study was "fun".

The curriculum was generally consistent throughout the final run of the study, but inevitably curriculum errors and simulator bugs were uncovered as more students participated. When a curriculum error came up (such as an incorrect figure), an observer would tell the student what the correct unit should have looked like, and the curriculum would be corrected for the next run. Occasionally simulator bugs would require a simulator restart. We believe that neither factor affected our results.

V. PHASE III: EXPANDED HUMAN COMPARISON STUDY

A. Study Objectives

Our goal was for this case study to provide us with a set of benchmark lessons and tests by which to measure an automated student's ability to learn from an automated teacher. While there were no formal requirements for human performance in this phase of the program, based on our experience with the Phase II evaluation and the Phase III curriculum, we again aimed to produce lessons and tests on

which human students who scored less than 20% on a pre-test could achieve a post-test score of at least 75-80%.

In this phase, the range of instruction methods used in the curriculum and the difficulty and complexity of tasks given to computational agents increased (over Phase II). Accordingly, we broadened the human student curriculum, increased the total number of human subjects from 28 to 75, and increased the number and diversity of treatment groups. In addition to benchmarking subjects given all three NIM types, we also benchmarked students given only a single NIM, students given only two NIMs, and students who were allowed to ask the study supervisor questions (through a restricted interface). We will refer to subjects who received all three NIM types as the *baseline* treatment.

B. Study Design

For Phase III, we added new learning challenges to the curriculum as well as a number of *Relaxation Trajectories*, which are changes to the curriculum designed to explore the ability of e-students to deal with more difficult aspects of human instruction. For example, in Phase II precise terms were used in the vocabulary of the lessons, and the e-student was given certain pieces of basic information about each lesson, including which NIMs were being used and the names of the procedures being taught. This initial approach was a result of some of the lessons learned in the Phase I Blocksworld study [3,4]; in that study, the human teachers had to be very precise in their bottom-up instructions in order for the e-student to understand the lessons being taught.

In Phase III we greatly streamlined and automated our testing setup. Whereas in Phase II we could test at most two subjects at a time, we now have the ability to test up to six subjects at once (Fig. 4). We assigned each subject to a station in our testing lab, each equipped with a Windows PC with their particular curriculum pre-loaded.

Each of these workstations mounted a Network Attached Storage (NAS) device containing a specially generated curriculum folder for each workstation. The corresponding curriculum folder was linked on the desktop of each machine, making all curriculum materials easily accessible to the subject.

We also mounted the NAS on our Linux server. This allowed the study supervisors to run our curriculum-generating scripts on the server and automatically create and distribute new curriculum folders to each workstation in preparation for each study group.

To ensure that subjects followed proper protocol, study supervisors monitored subject progress in person (by walking around and observing) and through monitoring scripts set up on our server. We used automated grading for tests, and the domain simulator provided the ability to replay tests if there was any question as to what actions a subject performed.

C. Participant Selection and Scheduling

The study participants were volunteer respondents to mass emails we sent to the Electrical and Computer Engineering, History, and Public Affairs graduate-student mailing lists at

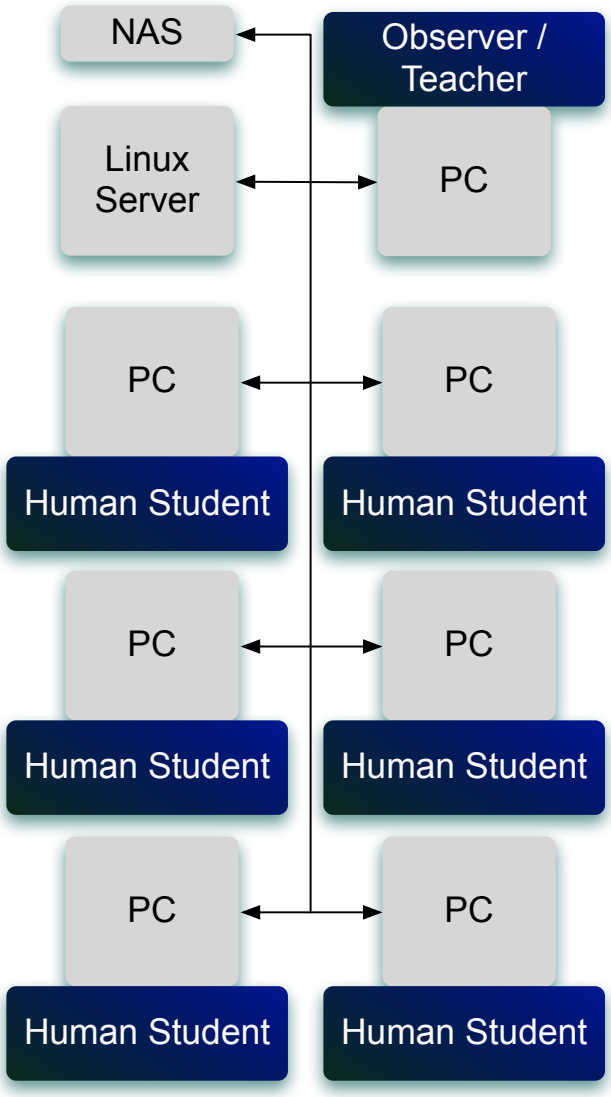

Fig. 4. Phase III: Lab Setup (for six parallel subjects)

The University of Texas at Austin. Potential subjects were asked to sign up for our study mailing list (through Google Groups), and we announced study times on this list. Those who qualified were offered $75 if they completed the study or $7.50 per hour if they withdrew or were unable to finish. Participants were told the study would take approximately three hours, and that they would have a maximum of four hours to complete the study.

We used the website Doodle.com, a site designed to help groups agree on dates and times for events, to handle signups for our study. When we decided on a study time, we posted the time to our account on Doodle.com, sent the signup-link to our potential-subject mailing list, and Doodle.com allowed up to six students to sign up for any particular study time (first-come first-served), recorded their name and email address, and sent us the information. Information about students who signed up

for studies was visible only to us (the study supervisors), not to other subjects.

We tested a maximum of six subjects concurrently, and we held at most two testing sessions per day. We varied our study times to accommodate subject availability, including morning, afternoon, evening, and weekend study times.

D. Benchmark Procedure

In our email inviting students to sign up for a study session, subjects were provided documents giving a broad overview of how the study would proceed and basic instructions for when they arrived. The approximate schedule for a participant was the same as in Phase II (Table I).

As each subject arrived, they were assigned a workstation and required to fill in their major and student status (undergraduate/graduate/other) on a *Subject Data* form. The study supervisor would then collect this form and fill in the date and Subject ID fields. This form is much shorter than its counterpart in Phase II, since much of the data that was taken by hand in Phase II was automatically recorded in Phase III.

After all subjects had arrived and completed their forms, the instructor would record the study *Start Time* and begin the 15-minute background knowledge presentation. This was not part of the curriculum we were evaluating; this was knowledge that human subjects needed to know before beginning the true curriculum and mostly consisted of an overview of the hidden domain simulator GUI. The e-student does not use the GUI interface. No material on diagnosing or fixing problems with the satellite-tracking rig was covered by this presentation.

Then, the study supervisor would have each student open their study navigation page (a simple HTML page linking to each curriculum component in order) and begin their self-paced study, beginning with the pre-test. The pre-test, consisting of a single randomly chosen fault, was meant to determine if a student had any previous knowledge of the domain. The only possible scores were 0% or 100%, and a student only scored 100% if he completely fixed all problems within the specified time limits. We excluded students who scored 100% from our post-experiment analysis. The amount of time given to subjects and the particular fault scenario chosen varied by experiment.

After taking the pre-test, students continued with the regular curriculum at their own pace, taking quizzes in the order indicated on the navigation page. The study observer could view students progress through a monitoring program on our Linux server (which monitored quiz and test log files), or by walking around and looking at their screens.

Except for in the Question and Answer (Q&A) treatment (Section V-H), the observer was not allowed to answer any questions about the material itself, so students were forced to rely on the curriculum for all of their domain knowledge. Five multiple-choice quizzes were administered at fixed points in the lessons that tested students understanding of previous units. In the main treatment, where students received all three NIM types, students were not allowed to continue on to further lessons unless they scored a minimum of 80% on each quiz.

In the single- and double-NIM treatments we assumed that subjects would do poorly, but we wanted to see how they would do on the full set of rungs anyway. Therefore in these treatments, we configured the quizzes to not reveal scores, and we did not require subjects to pass the quizzes (though we did not tell them that).

After a student successfully completed all the lessons and quizzes, they continued on to the post-tests. All post-tests consisted of the student diagnosing and fixing problems with the satellite-tracking rig in the domain simulator. The amount of time given to subjects and the particular fault scenarios chosen varied by experiment.

E. Experiment Configurations

For the pre-test in our first experiment, a single fault scenario was chosen out of six possible (see Table III). The *Null Fault* and *Multi Fault* scenarios were excluded. The student had eight minutes to diagnose and fix any problems with the satellite-tracking rig: one minute for each state of the domain simulator.

Two post-tests were given. First, a student was given five different at-most-one-fault scenarios, randomly selected out of seven, administered in a random order (see Table III). The *Null Fault* case was one of the possible scenarios for this test. Afterwards, some students were given a *Multi-Fault* scenario, in which the satellite-tracking rig had several different faults in a single run. In each post-test fault scenario, the student had eight minutes to diagnose and fix any problems.

To add realism to the test scenarios, in this initial experiment we configured the test simulator to advance through states in real-time rather than allowing the subject to manually advance the states. This configuration caused unexpectedly low post-test scores; students were averaging 57% on the post-test as opposed to the Phase II mean of greater than 90%. Some reduction in post-test scores was expected, since this Phase's curriculum was harder than Phase II's, but we did not expect a reduction this severe. We attributed the reduction to the real-time clock for two reasons:

1) **Training vs. Knowledge:** Though the overall test-scenario time was greater than in Phase II (8 minutes rather than 5 minutes), the critical time-window in which subjects were required to perform each task was shorter (1 minute rather the entire 5 minutes of Phase II). We originally thought this indicated that success in real-time scenarios was more a matter of training (improving a skill through repetitive practice) than knowledge, as students scored much better when the real-time constraint was removed. This is discussed further in Section VI.
2) **Boredom:** Since subjects were not allowed to manually advance the simulator past states in which nothing occurs, we observed them losing focus while waiting through these states. Subjects were clearly less attentive and energetic when critical simulation events came to pass.

Additionally, one test scenario (Antenna Intrack Pointing Error) seemed to be too difficult in our initial setup. In this scenario, the student was required to take multiple actions

TABLE III
PHASE III POST-TEST FAULT SCENARIOS

Fault Scenario	p3.realtime		p3.non-realtime	
	pre-test	post-test	pre-test	post-test
Antenna Motion Error	s	s	s	s
Baseband Misconfiguration	s	s	s	s
Antenna Intrack Pointing Error (Easy)			s	s
Antenna Intrack Pointing Error (Hard)	s	s		A
Antenna Azimuth Pointing Error	s	s	s	s
Antenna Elevation Pointing Error	s	s	s	s
Baseband Hangup	s	s	s	s
Null Fault		s		s
Multi-Fault		s		A

s - Given to some students in this treatment
A - Given to all students in this treatment

instead of just one as in the other scenarios; students often either missed or ignored one of the necessary actions. In Table III, we refer to this scenario as *Antenna Intrack Pointing Error (Hard)*. In our second, revised experiment, we replaced it in the main post-test with an easier version (*Antenna Intrack Pointing Error (Easy)*) and gave all students the *Hard* version afterwards.

Also, in our revised experiment we returned to allowing subjects to advance the simulator by hand, and gave students a total of five minutes for each scenario (as in Phase II). Average final test scores for the baseline condition jumped from 57% to 81%, validating our hypothesis that a large part of the score reduction was due to the real-time clock.

F. Baseline Results

A total of two subjects were used in preliminary dry runs, both of whom were Graduate Research Assistants under Professor Perry. 20 subjects participated fully in our initial study design (real-time clock), and one additional subject chose to leave the study early. 55 subjects participated fully in our revised study design, and one additional subject chose to leave early. In all results we exclude students that passed the pretest (5 students total in Phase III). When we discuss a *post-test score* in the following analysis, we mean the fraction correct out of the five randomly-chosen post-test scenarios. We discuss the *Multi-Fault* and (in the case of p3.non-realtime) *Antenna Intrack Pointing Error (Hard)* scenarios separately.

We can compare three groups of *baseline* subjects: those from Phase II (denoted p2.non-realtime), those in Phase III with a realtime test clock (denoted p3.realtime), and those in Phase III without a realtime test-clock (denoted p3.non-realtime). These groups contained 28, 12, and 19 subjects respectively. As shown in Fig. 5, all baseline subjects completed the study in under the four-hour time limit. Because of the curriculum extensions, the Phase III subjects generally took longer than the Phase II subjects. The median p3.non-realtime study time is slightly longer than the median p3.realtime study time possibly because of the additional post-test scenario.

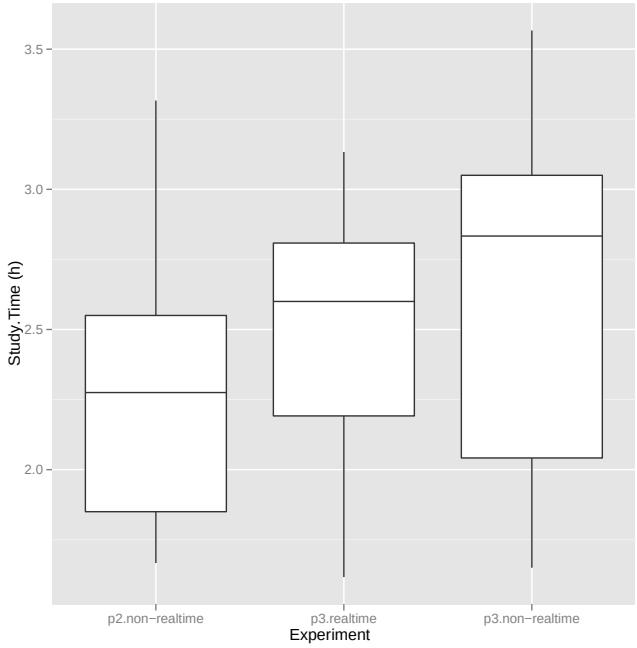

Fig. 5. Study Completion Time for Baseline

The mean post-test score for p3.realtime students was 57%; for the p3.non-realtime students, the mean was 81%, surpassing our goal of 80%. The median scores for all three baseline groups are shown in Figure 5. It is clear from the figure that our curriculum changes between the two Phase III groups improved scores across the board, not just the mean. The Phase III curriculum overall is more difficult than the Phase II curriculum, and our p3.non-realtime curriculum achieved our goal of 80% while being more discriminative than the Phase II curriculum. The p2.non-realtime box plot is compressed to a line because the interquartile range (IQR) is zero; most of the post-test scores were 1.0, including the 25th percentile, the median, and the 75th percentile.

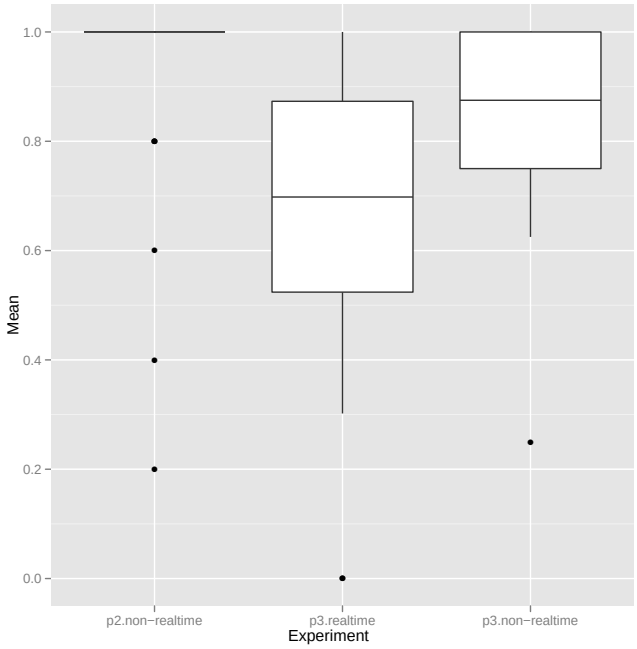

Fig. 6. Post-test Scores for Baseline

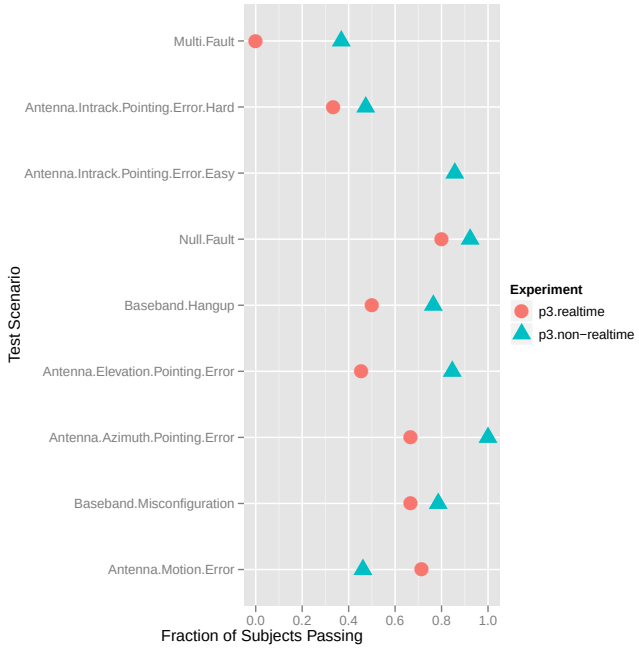

Fig. 7. Baseline Post-test Scores for Baseline

We can also compare the means for each post-test scenario. In Fig. 7, the p3.non-realtime scores almost dominate the p3.realtime scores, as expected (excepting only *Antenna Motion Error*). Additionally, the jump in passing rate between *Antenna Intrack Pointing Error (Hard)* and ... *(Easy)* is clear in this plot. p3.non-realtime does not have the *Antenna Intrack Pointing Error (Easy)* scenario since the Easy scenario was created in response to it. We do not compare against the p2.non-realtime means here, since many of the Phase II fault scenarios have been changed enough to no longer be directly comparable.

Finally, we compare the progression of quiz scores among the three baseline groups (first tries). The median score for all first-try quizzes in all experiments is 100%, and the inter-quartile range of the quiz scores decreases to 0 as the subjects progress through the quizzes, indicating they are learning.In Table IV, we provide the means in tabular form for comparison. Since means are more affected by outliers, the story is not as clear.

G. Other Results

In addition to the baseline, in Phase III we also tested subjects with only two NIMs, and subjects given only a single NIM. We restrict this analysis to the p3.non-realtime curriculum, since in p2.non-realtime we only had the baseline treatment, and in p3.realtime we only tested a few non-baseline subjects before decided to switch to a non-realtime testing procedure. We denote the NIMs as T (*by telling*), E (*by example*), and F (*by feedback*). In these results, we group in the few subjects allowed to ask questions, since in our study they ended up asking so few questions.

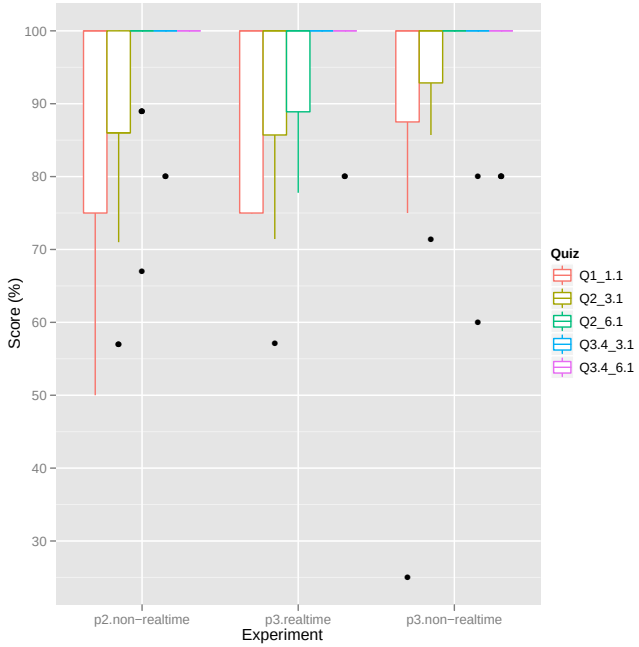

Fig. 8. Baseline Quiz Scores

In Fig. 9, we show the length of study by NIM-set given. We can see that subjects tended to take less time when given the T NIM, and more time when given the F NIM. In general, the T lessons were shorter than the F lessons, and the F lessons also required subjects to interact with the simulator and encouraged subjects to repeat if necessary.

In Figure Fig. 10, we show mean post-test score by NIM-

TABLE IV
BASELINE MEAN QUIZ SCORES (%)

	Q1_1.1	Q2_3.1	Q2_6.1	Q3.4_3.1	Q3.4_6.1
p2.non-realtime	89.3	88.8	97.6	98.6	100.0
p3.realtime	90.9	89.6	95.0	100.0	96.4
p3.non-realtime	90.8	95.5	100.0	96.8	96.8

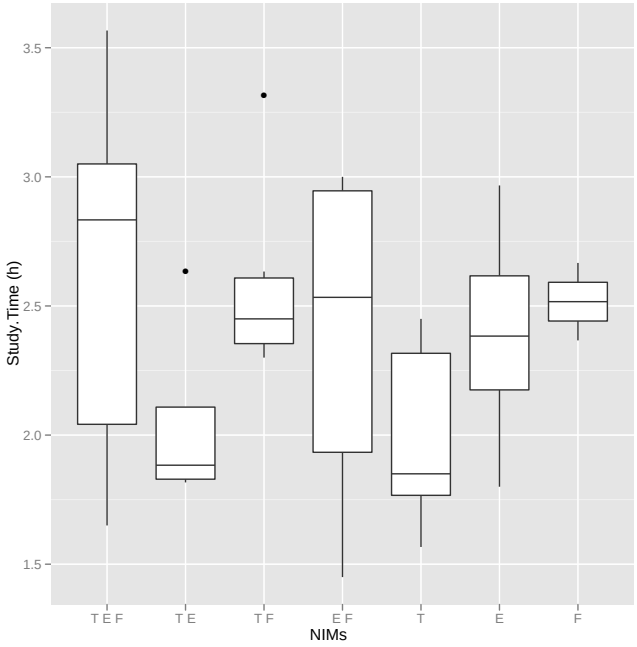

Fig. 9. p3.non-realtime Study Times by NIM

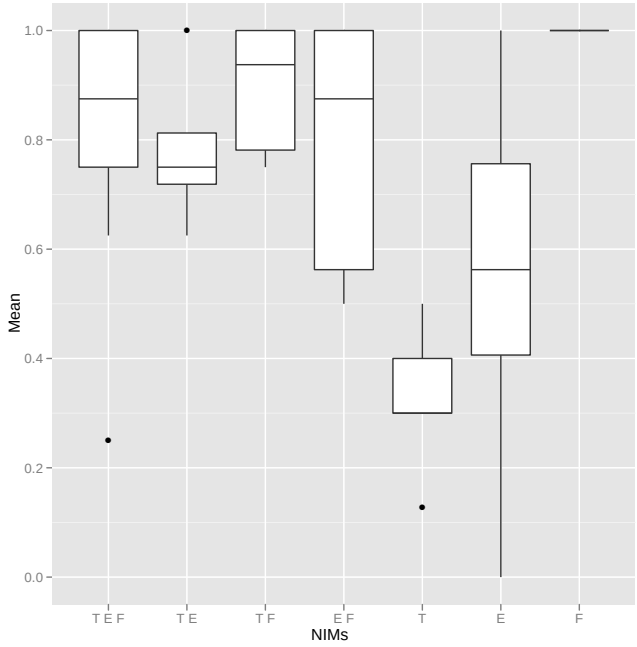

Fig. 10. p3.non-realtime Mean Post Test Score by NIM Set

set. The double-NIM subjects seemed to do almost as well as those given the full set of NIMs. The subjects given the F NIM seemed to do the best of the restricted sets; in fact, all subjects in the F-only group had a perfect post-test score. There are several reasons why this might be the case. This NIM is somewhat a combination of *by feedback* and *by telling*; in the self-paced *by feedback* lessons, if a subject doesn't follow the correct procedure, the subject is given a by telling description of what should have been done. This is also the only NIM where subjects get any practice with the simulator before the post-test.

H. Q&A

We also tried allowing subjects of the lowest performing NIM sets (T and E) to ask questions through a restricted interface. Counter to our expectations, it was very difficult to get subjects to ask anything. Finally, by choosing subjects who knew us and who were in relatively non-technical fields, we got a few questions. Even then, the questions were relatively basic questions about definitions, such as, "what does it mean to 'input to a component' ", and a misunderstanding about "absolute value".

VI. LESSONS LEARNED AND RECOMMENDATIONS

Creating the context and protocols needed to evaluate learning software agents has been a challenging enterprise. It is our hope that with our procedures and recommendations, researchers can benchmark the performance of human subjects on a domain of choice and confidently use this benchmark to evaluate agent performance.

A. Timing of Tests

In our results students, did much worse in p3.realtime than in p2.non-realtime. We hypothesized that this was because passable performance required *training* (improving a skill through repetitive practice) rather than *knowledge*.

Was the drop in scores between p2.non-realtime and p3.realtime in part due to the difference between training and knowledge? That is, were we attempting to test their knowledge but instead testing a skill that would require a different sort of curriculum? Here is what Allen Newell and Paul S. Rosenbloom have to say on the subject in the book chapter, "Acquisition of problem-solving skill":

> Practice used to be a basic topic. For instance, the first edition of Woodworth (1938) has a chapter

> entitled "Practice and Skill." But, as Woodworth [p. 156] says, "There is no essential difference between practice and learning except that the practice experiment takes longer." Thus, practice has not remained a topic by itself but has become simply a variant term for talking about learning skills through the repetition of practice. [14]

This implies that no, there is no difference between knowledge (learning) and training (practice). However, time to complete tasks improves with practice at a rate well-modeled by a power law [14], and it is possible that students were not given enough time in p3.realtime given their level of practice.

In both p2 and p3, students learned which simulator states were likely to be significant. In p2.non-realtime, students would quickly advance past the unimportant states, leaving most of their 5-minute test time to do the essential diagnosis and repair. In p3.realtime, each state lasted 1 minute, so a student had to wait for the correct state and then perform diagnosis and repair in under a minute, so he was essentially given less time.

In future benchmarking experiments, experimenters should simply ensure that students are given a reasonable amount of time to complete each test task, whether they employ realtime testing or not. We are not testing human subjects, we are attempting to provide an appropriate human benchmark for e-student testing, so test time should be selected so that a requisite number of human subjects pass.

If the benchmarking is similar enough to ours, our testing times can be used as starting points. If not, reasonable testing times can be determined with pilot studies or generated by modeling the test with the Model Human Processor method [15] or GOMS [16].

Should realtime testing or non-realtime testing be used? That should be decided based on the importance of this issue to the internal validity of the domain being studied balanced against the impact of boredom. In our domain, since boredom was a real issue in human benchmarking and realtime testing was deemed nonessential, we went with non-realtime testing.

B. Other Lessons Learned

- Human and electronic students differ in fundamental ways that make it difficult to create analogous contexts without providing one side with undue advantages over the other.
- Seemingly insignificant semantic details are critically important in the attempt to provide analogous contexts.
- The self-pacing mechanism for human teaching and learning proved an invaluable insight in establishing the e-student benchmarks.
- Increasing automation of lesson structures has been essential for the extensions we needed for Phase III.

ACKNOWLEDGEMENTS

This research was sponsored by AFRL under contract FA8650-07-C-7722. Special thanks to our colleagues at BAE Systems, Cycorp, Inc., Sarnoff Corporation, Stottler Henke Associates, Inc., and Teknowledge Corporation for their collaboration in developing the various BL curricula and testing materials, to Matthew Berland, David DeAngelis, and Dan Luu for earlier work on this project, and to study participants at UT.

REFERENCES

[1] D. Oblinger, "Bootstrapped learning," Presentation, 2006.

[2] R. D. Grant, D. DeAngelis, D. Luu, D. E. Perry, and K. Ryall, "Designing human benchmark experiments for testing software agents," in *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering*, M. Niazi and M. Turner, Eds. Durham, UK: BCS eWIC, April 2011, pp. 124–128.

[3] ——, "Towards evaluating human-instructable software agents," in *Proceedings of the IADIS International Conference Interfaces and Computer Human Interaction*, Rome, Italy, July 2011.

[4] M. Berland and D. E. Perry, "Novice human teachers of a virtual toddler: A case study," The University of Texas at Austin, Tech. Rep., 2009, http://users.ece.utexas.edu/~perry/work/papers/090123-MB-blexp1.pdf.

[5] N. Shachtman, "Darpa wants software students," http://www.wired.com/dangerroom/2007/08/can-you-teach-a/, August 2007.

[6] J. Ludwig, J. Mohammed, and J. Ong, "Developing an international space station curriculum for the bootstrapped learning program," in *Aerospace Conference, 2010 IEEE*, March 2010, pp. 1 –8.

[7] J. Ludwig, A. Davis, M. Abrams, and J. Curtis, "Developing a hidden domain for human and electronic students," 2011.

[8] C. Morrison, D. Bryce, I. Fasel, and A. Rebguns, "Augmenting instructable computing with planning technology," in *ICAPS'09 Workshop on the International Competition for Knowledge Engineering in Planning and Scheduling*, 2009.

[9] S. Natarajan, G. Kunapuli, R. Maclin, D. Page, C. O'Reilly, T. Walker, and J. Shavlik, "Learning from human teachers: Issues and challenges for ilp in bootstrap learning," in *AAMAS Workshop on Agents Learning Interactively from Human Teachers*, 2010.

[10] T. Kaochar, R. Peralta, C. Morrison, T. Walsh, I. Fasel, S. Beyon, A. Tran, J. Wright, and P. Cohen, "Human natural instruction of a simulated electronic student," in *AAAI Spring Symposium*, 2011.

[11] J. Beal, P. Robertson, and R. Laddaga, "Curricula and metrics to investigate human-like learning," in *AAAI 2009 Spring Symposium Agents that Learn from Human Teachers*, 2009.

[12] J. Beal, A. Leung, and R. Laddaga, "Spectrum curricula for measuring teachability," in *AAMAS 2010 Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2010.

[13] J. Davis, K. Leelawong, K. Belynne, B. Bodenheimer, G. Biswas, N. Vye, and J. Bransford, "Intelligent user interface design for teachable agent systems," in *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM, 2003, pp. 26–33.

[14] J. Anderson, J. Greeno, P. Kline, and D. Neves, *Cognitive skills and their acquisition*. Lawrence Erlbaum Associates, 1981, ch. 1, Acquisition of problem-solving skill, pp. 191–230.

[15] S. Card, T. Moran, and A. Newell, "The model human processor- An engineering model of human performance," *Handbook of perception and human performance.*, vol. 2, pp. 45–1, 1986.

[16] B. E. John and D. E. Kieras, "The goms family of user interface analysis techniques: Comparison and contrast," *ACM Trans. Comput.-Hum. Interact.*, vol. 3, no. 4, pp. 320–351, 1996.