A Reconfigurable Architecture for building Intelligent Learning Environments

Joseph G. Linn, James Segedy, Hogyeong Jeong, Benjamin Podgursky, and Gautam Biswas
Dept. of EECS/ISIS, Vanderbilt University, Nashville, TN. USA.

Abstract. This paper describes our initial efforts at implementing a new Choice-Adaptive Intelligent Learning Environment (CAILE) that combines multi-agent adaptive technologies and service architectures to provide a framework for designing extendible and reconfigurable learning environments. We describe the core components of the CAILE architecture, learning tasks that establish a situated context for learning, and a set of customizable agents that support student learning. We employ software engineering metrics to evaluate the system, and illustrate the reconfigurable and extensible properties of our design and implementation.

Keywords. Intelligent Learning Environments, Choice-Adaptive Intelligent Learning Environments, Multi-agent architectures

Introduction

The last decade has seen many technological innovations in computer-based learning environments. New technologies can seamlessly combine state-of-the-art simulation engines, multi-agent architectures, on-line collaborations, and multi-media functions to support pedagogies derived from behaviorist, cognitive, and socio-cultural theories of learning [2][3][4][10][17]. Of particular interest to us are learning environments that provide students with a rich set of choices on how to learn by including many different learning interactions and resources. The idea is to help students become adept at making appropriate choices that benefit their own learning, and in turn these students will have the opportunity to develop learning strategies and metacognitive skills that will make them more effective learners when they leave the structured curricula characteristics of much school-inspired instruction.

However, there are several challenges in designing configurable learning environments that promote the development of self-regulated learning strategies. One challenge is that choice of learning activity can confuse and distract inexperienced learners, which may adversely affect their learning abilities [6]. Another challenge is that learning environments today mostly consist of a large number of scattered and incompatible designs. Each new research idea in the area of computer-based learning environments generally requires the costly design of a new system or the lengthy redesign of an existing system in order to test that idea.

Given these challenges, our goals are two-fold: we aim to (1) facilitate the implementation of agents that intelligently adapt to student choices to help guide them toward more effective metacognitive strategies; and (2) design a generic framework and associated platform that allows researchers to quickly and efficiently design, imple-

ment, and re-implement learning environments that address their research questions. We believe that we have made meaningful progress towards achieving these goals with our Choice-Adaptive Intelligent Learning Environment (CAILE) framework. This novel framework combines multi-agent adaptive technologies with service architectures [18] to provide a generic framework for reconfigurable learning environments. In this paper, we review our past work on designing learning environments and propose a new set of design principles to create a generic framework for building CAILEs. We then discuss related work that proposes methods for helping students learn how to make choices. Next, we discuss the design and implementation of this new CAILE architecture, including suggestions on how to re-implement our Teachable Agents system. Finally, we propose future directions for this line of research.

1. Past Work, Related Work, and a New Set of Design Principles

Our original Teachable Agent (TA) architecture [20] was agent-centric and designed to include a number of agents that communicated via an environment platform. The student, also represented as an agent in the system, learned by teaching and interacting with the computer agents. The environment platform contained the system interface, and performed all of the functions for manipulating the interface, such as making changes to a map, displaying quiz results, and animating the concept map as the TA explained her answer. The agent design adopted a hierarchical object-oriented structure with a generic agent parameterized by its appearance and its behavior (including what it monitored in the environment and how it responded to events in the environment).

The agent-centric architecture worked well initially because our TA systems used a few agents with a fixed set of functionalities. However, attempts to build on this architecture for new studies that asked new research questions, became difficult because of our old design decisions produced a lack of flexibility. The new studies required considerable effort and time for rewriting code. More specifically, we encountered the following three problems: (1) the logic for services and agent feedback were tightly coupled, yet scattered throughout the code, (2) agent behaviors became harder to configure, and (3) the code progressively became more difficult to understand.

Other systems also support self-regulated learning strategies. AutoTutor [13] uses an agent and latent semantic analysis to simulate a natural tutorial dialog with a student. As part of a question-answer process, the agent encourages students to think deeply about the material they are learning by asking for more details and better explanations, and adaptively correcting students' errors. HelpTutor [1] takes a more direct approach by informing students on when and how to ask for help in a traditional ITS framework.

Animated pedagogical agents have extended tutoring and coaching opportunities by using life-like animated characters that interact with students in virtual problem solving environments using hypermedia, and human-like gestures and emotions. The notion of agents as tutors or coaches has been extended to learning companions [10] that assist, critique, and motivate during problem solving. Example systems include EduAgents [16], LECOBA [21], and Active Worlds [17]. Chen, et al. [8] have employed the idea of nurturing virtual animal pets in open learner environments as a motivational tool for learning. These systems, in addition to our own, are validated by literature that suggests that giving students choices in terms of learning activities as well as the nature of feedback they receive, can, in the right contexts, lead to intrinsic motivation [22]. Thus, the value of choice-driven systems comes both from its potential to

motivate students as well as the ability to provide students opportunities to learn how to make better choices in future learning.

Building on these past systems, our new CAILE framework incorporates a number of design principles. First, the CAILE framework focuses on standardizing the monitoring tasks in the system in order to better characterize student behaviors and respond to them. Second, the CAILE framework makes a conscious effort to modularize the system architecture, so that an agent's behaviors are distinct from the services that compose the learning task. Finally, the architecture is constructed to allow for future expansions that will allow agent behaviors to be designed hierarchically and in graphical environments that are more accessible than inline computer code. This is an important step in allowing cognitive scientists and educational researchers to better guide the system development process.

2. Components of the System Architecture

Our approach to designing intelligent learning environment encompasses: (1) a set of *learning tasks* for the students; and (2) *agents* that help guide the student through the task. CAILE includes an Environment Platform, which facilitates the interactions of the different pieces; however, a complete description of the platform has been omitted here for space considerations. Our architecture, illustrated in Fig. 1, implements the learning tasks and agents in a modular and extensible fashion by providing libraries of functions, which may be rapidly configured to work together, updated, and extended to enable different kinds of learning tasks along with different learning strategies.

Learning tasks correspond to situated units that define the content and context for learning (e.g., [5]). A learning environment may contain one or more learning tasks. Learning tasks are themselves constructed from a set of modular services. Services may or may not have a graphical component. Some services are used by the agents to analyze the student's and other agents' activities in the system. These services are explicitly linked to agent behavior in a way that tailors the behavior to the current task. The key elements of a service are that services perform functions (e.g., help an agent reason), and they relay information to the agents (e.g. the users' actions in the system). Agents can command a service make changes in the system environment.

Much like web-based service-oriented architectures [15], the set of services are organized into a *Service Library* to support the rapid inclusion of different task interfaces and other components. A rich service library allows a learning environment designer to quickly configure learning tasks by combining service components in meaningful ways, and thus build a number of different environments with very little additional programming effort.

Agents in our CAILE systems are animated characters that interact with human users to support instruction and scaffolding. They have assigned roles (e.g., tutor, tutee, or peer), which provide the social component to our learning environments. The benefit of the CAILE architecture is it can be rapidly expanded to include new agents to support the scaffolding needs of current learning tasks.

An agent's behavior is defined by its response to chosen patterns derived from events associated with services and agent actions. In order to implement a desired agent role, agents are configured to respond to specific patterns of events, which are interpreted as behaviors in the context of learning tasks. Lower-level events can be aggregated to form higher-level events. At the lowest level, the events are generated by ser-

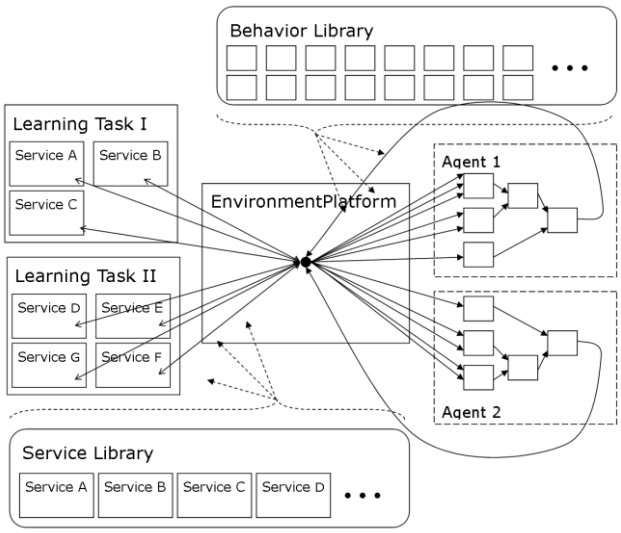

Figure 1. Choice Adaptive Intelligent Learning Environment Architecture.

vices with a graphical component (e.g., button clicks). As a session evolves, analytical services and even agents' behaviors can form the basis for defining more complex behaviors. This creates hierarchical structures that are illustrated within the agents in Fig. 1.

The Behavior Library stores configured agent behaviors. The behaviors are derived from the set of services in the learning tasks and the roles assigned to the agents in the environment. For example, a Reading Mentor can include behaviors that monitor student reading activities and provide reading strategy feedback. Like the service library, the behavior library is a repository for different behaviors that can be defined from services and associated events, and then used to create agent behaviors.

Configuring the system to include sets of behaviors and support services that are organized into the two libraries gives the learning system designer the ability to quickly assemble a large number of possible systems in different configurations to allow for various learning tasks, and then easily design and tailor agents to provide different pedagogical roles in the learning environment. We have developed XML templates to facilitate these configuring and rebuilding tasks. Building on the XML configuration approach, a key element of future CAILE systems will include the ability to reconfigure themselves "on the fly" to adjust to the student's learning styles and interaction patterns.

3. The Event Architecture

As discussed, events mark actions in the system by the user, the virtual agents, and the services. This necessitates having system components that manage the complexity of communicating all of this information. The CAILE architecture uses a low-level communication layer called the *event architecture*, which is a message-passing system for services and agents.

3.1. Events

The redesigned agent framework needed a new communication protocol to enable agents to track the student's and other agents' activities in the system. The event-based system allows state information to be packaged with the communication object. To

accomplish the specific forms of communication in the architecture, we identified three types of logical events: *service events*, *agent events*, and *internal events*.

Service events are generated by elements of the learning task, reporting student actions, e.g., a student just clicked a button. Agent events, on the other hand, are generated in response to patterns of service events, and they invoke a service function. Internal events map low-level patterns to higher-level behaviors. Figure 1 shows that these behavior outputs do not leave the agent space.

3.2. Defining Agent Behavior

A *rule*, the fundamental unit that defines agent behavior, is a pattern identifier which is configured using events. Rules can be designed to monitor the quantity and frequency of service events, agent events, and internal events that take place in the system. When a rule is triggered by its target behavior, it initiates a sequence that leads to the creation of a new event that either produces an action, or it gets aggregated into a higher-level internal event. Rules and internal events together define the logical structure of agent behavior, which governs their actions in the system.

In designing rules, it is necessary for the designer to specify the events and patterns the rule will detect. A rule designed to respond to student actions in a service related to the construction of a causal concept map cannot be applied to a similar service, say construction of a graph. Thus, when a service changes, so must the rules and behaviors directly associated with it. In building a learning environment, a designer must first choose the services that will make up the learning task from the service library, before choosing the agent behaviors from the behavior library that match these services. While this tight coupling might seem like a weakness of the CAILE architecture, it allows for a loose coupling between agent behaviors and service functionality. An agent's immediate behaviors are dependent only on the learning task. This allows researchers and system designers to swap agent behaviors in a modular way, which, in turn, affords them the ability to create experiments that test different agents and agent roles without requiring changes to the service operations. Also, higher level strategies are built hierarchically. If the higher-level events keep the same names, only the rules defining the lower-level events have to change to correspond to a changed service.

4. Redesigning Betty's Brain in the CAILE architecture

The CAILE architecture is general, but as a proof of concept we have reimplemented Betty's Brain [6] in the CAILE paradigm to take advantage of its configurability and lower development costs. In Betty's Brain, the teachable agent is the "learner". The student teaches Betty by building a causal concept map. She responds to students' queries and takes quizzes when asked. Occasionally, she makes comments about her learning progress to make the students aware of how well they are teaching her. Meanwhile, the Mentor agent plays the role of giving students more explicit feedback on their teaching practices and self regulated learning strategies.

Services form the basis for implementing this functionality. The core learning task in Betty's Brain is building a concept map of a domain. Examples of services to support this task include the concept-mapping tools, animation schemes, reasoning functions that apply to the concept map, the hypertext-based resources reader, and assessment tools. To support the teachable agent's role in this task, we implemented the

agent reasoning service so that the agent can answer questions using the concept map, and an explanation service, which animates the concept map as the agent answers a student generated question. The mentor focuses on assessment and teaching guidance so we have a quiz service, which grades the concept map by checking on how it answers a set of questions, and a behavioral analysis service to give directed feedback when the student asks the mentor for help. In addition to these underlying services, the agents also use a representation service to control their own image on the screen.

Once the services have been implemented, the first step in designing an agent behavior is to identify the appropriate interactions for an agent's role. This involves identifying the patterns of student behavior to which agents should respond. One such behavior is "guess-and-check", when students repeatedly quiz the teachable agent after making a single change to the concept map. To detect and respond to this behavior, a system designer can write rules for the Mentor agent that are directed to intervene with advice on alternative strategies. Using the services, the designer can determine which of the services generate the events that will define behaviors. In this case the events are "Request Quiz, Edit Map, Request Quiz," in sequence from the quiz and concept map services. The designer can then write a Rule that creates an internal event called "Guess and Check" which fires when the defined pattern occurs (the designer can also set reset events for good behaviors like checking the resources). From defined student behaviors, the researcher writes a second set of rules that creates Agent events. An example of such a rule firing would result in the mentor agent suggesting to the student to follow an alternative strategy. The mentor rule would be designed to create the agent event only when the "Guess and Check" internal event occurs quite frequently during the student's use of the program.

This example illustrates the potential power of the event architecture in allowing agents to observe and respond to students' individual behaviors. Additionally, it demonstrates how this can be extended to more complex situations, allowing for a powerful and rich set of agent behaviors in the system. This new implementation of Betty's Brain was evaluated using a set of software engineering metrics. The results imply that by following the CAILE framework, we have created a functionally richer Betty's Brain system that is actually easier to manage and extend than the predecessor systems.

5. System Evaluation and Discussion

As mentioned in the introduction, one of our main goals in developing the CAILE framework was to allow researchers to build computer-based learning environments more quickly without a lot of new code generation. To support this claim, we compared our old TA system with our new TA system that is re-implemented in the CAILE framework using the four software engineering metrics: (1) Lines of Code (LOC), the raw count of all lines of source code; (2) Weighted Methods per Class (WMC), the sum of the complexity of all methods in a class [9][9]; (3) Cyclomatic Complexity (CC), a measure of the complexity of the methods within a program; and (4) Lack of Cohesion of Methods (LCOM), which measures the extent to which classes in a program perform a single purpose [9].

Results obtained with the Eclipse Metrics plugin (http://eclipse-metrics.sourceforge.net) appear in Table 1. We evaluated three versions of the system: (1) the original version of Betty's Brain (2006); (2) an extended version that included support for self-regulated

learning strategies (2008); and the current implementation of Betty's Brain in the CAILE framework (2009).

Table 1. Comparison of different software versions of Betty's Brain using the specified metrics

Year	LOC	WMC	CC	LCOM
2006	16652	14.13	1.95	4.16
2008	25645	16.55	2.17	5.80
2009	15564	14.53	2.09	5.08

The results show that extending our system to perform additional complicated functions led to an increase in all four metrics, indicating that the code had become significantly more complex. However, the results also show that re-implementing the system in the CAILE framework eliminated most of the added complexity from the 2008 system without sacrificing any of its functionality.

In addition to these "behind the scenes" metrics of the source code, qualitative observations of the system in use have shown that our modular design of services and agent behaviors has allowed for multiple threads of execution, something that was not available in our old system. Because of this, agent behaviors and rules are executed in parallel and don't incur a noticeable cost in performance. The only potential drawback to this approach is a slightly longer load time, attributed to parsing the XML rules, but this added delay is on the order of seconds, and is a once-per-session cost, so the trade-off seems reasonable.

As this system has only very recently been re-implemented in the CAILE framework, an evaluation of its educational benefits in this context is currently unavailable. However, qualitative observations have shown that this system performs well in classroom studies, and overall, it has proved to be more robust than its predecessors. The reader is referred to [23] for a full discussion of the educational implications and evaluation of our system.

6. Conclusions and Future Work

We have proposed a novel framework for creating and maintaining Choice-Adaptive Intelligent Learning Environments. The goal is to provide students with more learning choice, and also allow researchers to flexibly design CBLEs that adapt to student behaviors, making it easier for them to run studies with multiple research questions.

As we move forward, we will add tools that facilitate design of different learning tasks and agents. We envision a "drag and drop" interface, with choices for configuring agent behaviors based on events from drop-down lists, which the tool has populated. These tools should also allow teachers to easily configure learning tasks for students, and, therefore, direct instruction towards how to make better choices in learning while also teaching domain material. While these are ambitious plans, we believe that our approach provides the flexible and extensible computational architecture for building the choice adaptive environments.

Acknowledgements: This work was supported by research grants from the Dept. of Education (R305H060089) and NSF REESE (0633856).

References

[1] Aleven, V., Roll, I., McLaren, B., Ryu, E.J., and Koedinger, K. (2005). An architecture to combine meta-cognitive and cognitive tutoring: Pilot testing the Help Tutor, *12th Intl. Conf. on AI in Education*, C.K. Looi, et al. (eds)., Amsterdam, The Netherlands, 17-24.

[2] Anderson, J.R., Boyle, C.F., and Reiser, B.J. (1985). Intelligent Tutoring Systems. *Science*, 228(4698): 456-462.

[3] Barab, S.A., Hay, K.E., Barnett, M., and Keating, T. (2000). Virtual solar system project: Building understanding through model building, *Journal of Research in Science Teaching*, 37(7):719-756.

[4] Blair, K., Schwartz, D., Biswas, G., and Leelawong, K. (2007). Pedagogical Agents for Learning by Teaching: Teachable Agents, Special issue of *Educational Technology on "Pedagogical Agents*," 47(1): 56-61, January.

[5] Brown, J.S., Collins, A., and Duguid, P. (1989). Situated Cognition and the culture of teaching, *Educational Technology*, 33(3): 10-15.

[6] Bonk, C.J. and Cummings, J.A. (1998). A Dozen Recommendations for Placing the Student at the Centre of Web-Based Learning, *Educational Media International*, 35(2):82-89.

[7] Bransford, J.D. and Schwartz, D.L. (1999). Rethinking Transfer: A simple proposal with multiple implications. Review of Research in Education, AERA, 24: 61-100.

[8] Chen, Z.H., Chou, C.Y., Deng, Y.C., and Chan, T.W. (2007). Active Open Learner Models as Animal Companions: Motivating Children to Learn through Interaction with My-Pet and Our-Pet, *International Journal of Artificial Intelligence in Education*, 17(3): 217-226.

[9] Chidamber, S.R., and Kemerer, C.R. (1994). A Metrics Suite for Object Oriented Design, *IEEE Transactions on Software Engineering*, 20(6): 476-493.

[10] Chou, C-Y., Chan, T-W., and Lin, C-J. (2003). Redefining the learning companion: the past, present, and future of educational agents, *Computers & Education* 40: 255–269

[11] Curtis, B. et al. (1979). , Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics, *IEEE Transactions on Software Engineering* , 5(2): 96-103.

[12] Dimitrova, V. (2003). STyLE-OLM: Interactive Open Learner Modeling, *International Journal of Artificial Intelligence in Education*, 13: 35-78.

[13] Graesser, A.C., Person, N., Lu, Z., Zeon, M.G., and McDaniel, B. (2006). Learning while holding a conversation with a computer, In L. Pytlik Zillig, M. Bodvarsson, & R. Bruning (Eds.), *Technology-based education: Bringing researchers and practitioners together*, Greenwich, CT: Information Age Publishing, 143-167.

[14] Graf, S. and Kinshuk. (2007). Providing Adaptive Courses in Learning Management Systems with Respect to Learning Styles. *In Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, Chesapeake, VA: AACE Press, 2576-2583.

[15] Hannafin, M.J. and Land, S.M. (1997). The foundations and assumptions of technology-enhanced, student centered learning environments, *Instructional Science*, 25: 167-202.

[16] Hietala, P. and Nierempo, T. (1998). The Competence of Learning Companions. *International Journal of Artificial Intelligence in Education* 9:178-19.

[17] Holmes, J. (2007). Designing agents to support learning by explaining, *Computers & Education*, 48: 523–547.

[18] Huhns, M.N. and Singh, M.P. (2005). Service-Oriented Computing: Key Concepts and Principles, *IEEE Internet Computing*, 9(1), 75-81, Jan/Feb.

[19] Jeong, H. and Biswas, G. (2008). Mining Student Behavior Models in Learning-by-Teaching Environments, *First International Conference on Educational Data Mining*, Montreal, R. S. Baker, T. Barnes, T., I.E. Beck, (eds.), pp. 127-136, Montreal, Quebec, Canada, June 20-21

[20] Leelawong, K., & Biswas, G. (2008). Designing Learning by Teaching Agents: The Betty's Brain System, *International Journal of Artificial Intelligence in Education*, 18(3): 181-208.

[21] Ramirez Uresti, J.A. and du Boulay, B. (2004). Expertise, Motivation, and Teaching in Learning by Teaching Systems, *International Journal of Artificial Intelligence in Education* **14**: 67-106.

[22] Ryan, R.M. and Deci, E.L. (2000). Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being, *American Psychologist*, 55(1): 68-78.

[23] Schraw, G. (2007). The use of computer-based environments for understanding and improving self-regulation. Metacognition Learning, 2:169–176.