Constructivism and the development of intelligent learning environments

Constructivist theories of learning present unique challenges to the instructional technology (IT) field requiring more responsive and adaptive computer environments.. Meanwhile, there have been a number of developments in the fields of human-computer interaction (HCI) and artificial intelligence (AI) that have potential in creating more "intelligent" learning environments. This paper will attempt to highlight some of the ideas of constructivism and the challenges they pose in multimedia design, and link them to some recent projects in HCI and AI which may answer some of those challenges.

Challenges of Constructivism

Constructivist theories of learning present a number of challenges to instructional technology (IT) design that should be addressed. Traditionally, instructional programs have been developed using behaviorist learning theories as a model, and thus greatly simplifying the technology required. Behaviorist systems typically consist of methodically instructing an individual based on a set of answers that are judged as right or wrong. Very little "intelligence" is required by either the computer or the student in these 'learning' scenarios. The implementation of constructivist theories, on the other hand, require more adaptive environments that can respond to the changing needs and whims of a student as he develops his own learning. Some of the learning activities spurned by constructivist ideas include discovery learning, apprenticeship, collaborative learning and proximal development (Duffy & Cunningham, 1998). Each of these areas can place unique demands on a learning program.

Discovery learning involves teaching a student the process of inquiry and learning. Traditionally, learning has been conceived as the transfer of knowledge from teacher to student. The knowledge that a student receives is an exact duplicate of that of the teacher. Discovery learning, on the other hand, relies on the idea that each individual constructs knowledge for themselves. In order for a student to truly comprehend the material, he must adapt the new knowledge to the cognitive structures created from past knowledge (Duffy & Cunningham, 1998). From the perspective of IT, the traditional mode of teaching facilitated simple learning designs. A program would have a finite amount of material that it had to provide to a student. It wouldn't be required to significantly adapt that information to the student's needs. Also, a program needed to merely test the input from students against the answers it has stored. No intelligent interpretation of the student's thought process is needed. A constructivist discovery learning program in contrast would have to respond to a student's inquiry needs. In a more complicated fashion, it would also be required to understand and interpret a student's responses, and match that to its own database.

Collaborative learning and apprenticeship are two significant learning tools derived from constructivist thought. Constructivists believe that knowledge exists in the environment (rather than in an individual's head). Social interactions between people are likewise essential to access this information. This collaboration in the constructivist process provides different perspectives and viewpoints. It allows students to access information that is not a part of their knowledge base. Collaborative learning also facilitates the knowledge acquisition through communication. Vygotsky—an early theorist whose ideas were antecedents to constructivism—believed that the very act of communication created knowledge structures in the communicator's head (Duffy & Cunningham, 1998).

Also, constructivists believe that learning is best accomplished in the context where that information is relevant. A learner must actively participate and engage in the setting that he wishes to study (Duffy & Cunningham, 1998). A student of politics, for example, should seek out a role in a state or national assembly. A student of zoology should look for opportunities to view animals in their habitats. This engagement in real-world settings also involves engaging with experienced individuals in that field. Who better to learn about Bolivian art than a Bolivian artist? Or if a student is interested in the technology that operates the space shuttle, shouldn't he model a NASA engineer? These relationships that students might form in these situations wouldn't be characterized as the typical student-teacher relationship. Rather than teaching by pouring the contents of one mind into that of another, the novice student learns by doing and interacting with others.

With regard to these learning tools-- apprenticeship and collaborative learning-- the Internet is an obvious tool which links a variety of individuals participating in a wide array of activities. It almost stands alone as a constructivist's tool in that regard. But if a student isn't skilled in the art of web searching, and is not certain what he is searching for, the Internet can quickly become more of a hassle than a help, and its potential benefit lost. Programs need to be able to return more relevant searches to match students' needs, and possibly recommend ideas that students had not yet thought of.

The zone of proximal development is another concept linked to the larger idea of constructivism. This idea refers to the difference between the actual level of knowledge or skills of a student and his potential level. It is within this range that learning can occur. If a student only works on things he already knows then he doesn't learn anything new. If he seeks to do things beyond his potential, the same holds true—he is unable to learn because he is incapable of understanding any of the concepts. If the student engages at a level above his expertise, but has the help of a teacher or mentor, this is the optimal level to achieve learning (Duffy & Cunningham, 1998).

In order to incorporate these concepts into an IT environment, programs need to be able to accurately judge the level of a student's knowledge and respond with the appropriate level of activities and information. Many programs—including the computer based GRE—have incorporated the concept of assessing a student's minimum and maximum performance level and making adjustments accordingly. These are quantitatively determined by calculation based on percentage right and wrong at certain levels. This may work for multiple-choice questions, but doesn't really apply in constructivist settings where the learner often determines his own unique answers to a problem. In order for a program to respond to such student input, a certain amount of qualitative assessment is also necessary.

Developments in intelligent learning environments

Generally, IT environments need to be more 'intelligent' if they are to respond to the constructivist learning needs of students. These intelligent learning environments need to be able to 1) understand the needs, motivation and thinking processes of students, and 2) dynamically respond to those needs in a relevant manner that causes students to reflect and deepen their understanding of the topic they're studying. Neither of these tasks are simple technological undertakings. The rest of this paper will present some projects that have attempted to address these issues and will be followed by an analysis and discussion of their applications.

Technologies that Connect: Watson and I2I

Watson is an intelligent information retrieval program developed at Northwestern University. It is one of a couple of new programs that attempt to make search engines more user-friendly. Watson is a unique program in that it monitors a person's work activities and proactively suggests supplemental material that the individual may want to incorporate into his documents. Watson is a supplemental program that could be used simultaneously with any word processing program or internet browser.

Watson actively logs everything the user does in an open program. Watson does not actually attempt to "understand" what the user is researching. Rather, it conducts a content analysis based on the frequency of "content-bearing" words, their position in the text, and their presentation attributes like type style and alignment. It uses this content analysis to conduct searches on the internet, intranets, multimedia libraries, news articles, and/or research databases. In order to not unnecessarily distract the user with useless, unnecessary information, Watson is selective in its searches, weighing the information and relevance to the user's topic, and filters out links whose relevance might not be apparent to the user. Also, the program presents its searches in an organized manner, using two axes of tabs to allow the user to select the type of content (images, news, documents, etc.) and the category of information (e.g. summary). Watson continuously updates its searches as a user works. It doesn't erase old searches in the process. Rather, every new search includes data from the past, so that each search represents the user's entire body of work (Budzik, Bradshaw, Xiaobin Fu & Hammond, 2002).

Watson has yet to be widely distributed, and is still being developed and tested. One test conducted on the program measured Watson's performance in searching for related websites using AltaVista and compared it to the performance of a group of expert-level searchers performing the same task. The human searchers returned queries that had 3 relevant documents out of every 10, while Watson returned 5 out of 10 relevant sites (Budzik & Hammond, 2000). Another study evaluated Watson's suggestions in the context of a task. Researchers submitted their last research paper for Watson to conduct searches on. The program returned two out of ten documents that the researchers found useful to their work (Budzik & Hammond &Birnbaum, 2000).

The results of these tests seem to be fairly remarkable. In both of these examples, Watson proves to be more adequate (or "intelligent") than its human counterparts in finding new material. The implication for education is positive. Watson uncovers information sources that students might not otherwise know thereby expanding their opportunities to grow their own knowledge.

I2I is a side-product of the Watson program and uses a similar architecture. I2I, though, has an extended focus: instead of connecting users to only artefacts in the online community, I2I connects its users to people based on the shared topics of their work content. At present, the Internet makes a wide array of people with a wide array of expertise available to anyone who can locate them. The problem, though, is taking the time to unearth these experts from the online communities of chat-rooms and list-serves. Often the reward for doing so doesn't seem worth the effort, particularly for smaller, short-term projects. I2I addresses this problem to further facilitate collaborative learning on the web (Budzik, Bradshaw, Xiaobin Fu & Hammond, 2002).

Like Watson, I2I conducts an analysis of every document a user activates or manipulates, running concurrently with any word processing or web browser program the user is using. It communicates with a *broker* which is located on a central server. The broker keeps track of different user ids, their open documents, and their contact information. When a user views a document, I2I uses information from the broker to list similar documents and other user ids that it has grouped with it (Budzik, Bradshaw, Xiaobin Fu & Hammond, 2002).

In this way I2I creates a more authentic browsing experience. If two people are viewing the same document at the same time, I2I would enable them to recognize that fact and initiate a dialogue about what they are viewing. The chances of this scenario, though, are often slim. So I2I allows users to track others who have recently viewed the site, and see what sites they are currently viewing. If a person wants to talk about a particular document, they can drop a "calling card" which other users would be able to see and respond to if they wish (Budzik, Bradshaw, Xiaobin Fu & Hammond, 2002).

I2I does allow for privacy. A user can shut off the tracking system at any time to allow for private viewing. A user's e-mail and identity is never released directly to others. Instead, all communication is mediated by a central server. From there, users can disclose their personal e-mail addresses at their own discretion (Budzik, Bradshaw, Xiaobin Fu & Hammond, 2002).

I2I is still in development and has some glitches to overcome. One factor that needs to be addressed is the threshold—or the sensitivity to grouping other documents and users in the same category. Tests have shown that popular sites tend to snowball in the number of visitors. The more people that view a site, the more that other users will see that it is a popular site, and got to it, thereby increasing exponentially the number of people that view ‘popular’ sites. In addition to risking server overload, these sites also tend to contain little pertinent content. This suggests that the I2I program risks introducing a “popularity-rules” structure to the Internet. Tests have shown that lowering the threshold will cause popular website viewing to diverge more quickly than at higher thresholds. Lowering the threshold, though, decreases the percentage of returned relevant results. As a result, determining I2I’s threshold level is a delicate balancing act which has yet to be resolved (Budzik, Bradshaw, Xiaobin Fu, & Hammond, 2002).

As for I2I’s implications on education, the program more easily facilitates collaborative learning. Students and academics who are working on similar projects would be able to recognize each other and share ideas about their work. The online world would develop more of a sense of an actual community as users could be tracked and recognized in the different websites that they visit.

Automated Learning Counterparts: AutoTutor

There have been several programs developed over the past two decades with the aim of providing students with an individualized tutor. One of the latest is AutoTutor-- a program developed at the University of Memphis designed to guide students to deepen their explanations and reflection of various academic topics. As the name suggests, the program creates an animated tutor who asks questions and follows up student’s responses with additional comments to elicit further information. Here is an example of part of an Autotutor session:

TUTOR: Suppose a runner is running in a straight line at constant speed, and the runner throws a pumpkin straight up. Where will the pumpkin land? Explain.
STUDENT: The pumpkin lands behind the runner.
TUTOR: Anything else?
STUDENT: I don't know.
TUTOR: Okay. What is the horizontal velocity of the pumpkin?
STUDENT: The pumpkin has a constant horizontal velocity.
TUTOR: so? The horizontal velocity of the pumpkin is the same as the?
STUDENT: Could you repeat that?
TUTOR: Once again, the horizontal velocity of the pumpkin is the same as the?
STUDENT: runner.
TUTOR: The same as the runner. Okay. Also, the man and the pumpkin have the same horizontal velocity. What can we say about the horizontal velocity of the pumpkin if there is no force acting on the pumpkin in the horizontal direction?
STUDENT: it's constant.

[Example from Graesser, Xiangen Hu, Susarla, Harter, Person, Louwerse & Olde-- http://www-2.cs.cmu.edu/~aleven/AIED2001WS/Graesser.pdf]

Each question that the tutor asks may have 3-5 expectations to elicit from the student. A single question may take up to 30 cycles to flesh out a question fully covering all of the expectations. The program uses a dialogue advancer network (DAN) to manage this conversational dialogue. The program doesn't judge the student's answers as right or wrong. Instead, AutoTutor weighs the response on a scale of completeness and incompleteness depending upon the number of points or expectations it incorporates (Graesser, Vanlehn, Rose, Jordan, Harter, 2001).

The DAN decides which expectation to address next and how to flesh it out. The dialogue it creates varies in directness and information content. The further along in the conversation the student goes with Autotutor, the more direct the computer's responses become and the more information is revealed. When a student's explanation has met a particular expectation, Autotutor moves on to another expectation. Autotutor uses "latent semantic analysis" to make this determination. A perfect match is scored as a 1 and no match is 0. An acceptable answer can be adjusted to be some score in between- .7 for example. When a student has met an expectation at this level the computer moves to the next criteria (Graesser, Vanlehn, Rose, Jordan, Harter, 2001).

Autotutor selects the order of expectations based on a couple of criteria. It selects material in which the student seems almost competent, but needs a bit more help, and then progressively moves to the more difficult issues. The program also takes coherence into account, attempting to make a logical segway between dialogue topics (Graesser, Vanlehn, Rose, Jordan, Harter, 2001).

The tutorial program provides different types of feedback. Periodically, the program's animated head nods and says, "uh, huh" to draw out a response from the student. This response has nothing to do with the correctness of the student's response; rather it is triggered when nouns are typed in. The program also provides evaluative pedagogical feedback. Based on the correctness of the student's previous response, the animated character will changes its expressions based on a scale of negative, neutral-negative, neutral, neutral positive, and positive. If a student replies with something completely off-base, the program will not attempt to make a connection between the student's response and its expectation. Instead AutoTutor will simply provide the answer (Graesser, Vanlehn, Rose, Jordan, Harter, 2001).

Autotutor has a number of features which make it seem intelligent. It judges a student's response qualitatively, rather than quantitatively. It attempts to manufacture dialogue and reactions as a real human would. Unfortunately its design isn't perfect, as this sample dialogue illustrates:

> **AutoTutor:** (Elaboration) And, the CPU uses RAM as a short-term memory.
> **Student**: to lab assistant [Should I, like, I can just say how I think he's unintelligible?]
> **Student**: types: I am not sure
> **AutoTutor** (Neutral Feedback) Uh-huh

AutoTutor: (Elaboration) Also, RAM's contents disappear when the computer is turned off.
Student: types what?
AutoTutor: (Neutral Feedback) Uh-huh
AutoTutor: (Hint) So, how does the CPU get information from RAM?
Student: types I am not sure
AutoTutor: (Neutral Feedback) Uh-huh
AutoTutor: (Summary) So, RAM is primary memory storage and is changeable. It is used by the CPU as short-term memory. The CPU both writes to and reads from RAM. The contents of RAM disappear when the computer is turned off.

[example from: Person, Bautista, Kreuz, Graesser, http://www.cogs.susx.ac.uk/users/bend/its2000/person.pdf.]

This sample dialogue demonstrates how Aututor is not at all cognizant of a student's needs. The technology in artificial intelligence has not reached the point where a computer can understand language that is out of the context of its programmed parameters. This inflexibility effectively prohibits students from engaging a computer mentor in a natural manner. The student's conversation and questions are limited to what the computer 'expects' the student to say. As a result, the ability of the automated tutor to facilitate intellectual growth is severely limited.

Affect Computing: A New Frontier

A new frontier connecting the fields of HCI and AI is the development of computer technology that is able to recognize emotion. Throughout much of the development of the field of AI, computer scientists have ignored the realm of emotions as "unpredictable and mysterious" (McNeese 2003). Early views of HCI viewed the mind as an independent box divorced from any factors in the environment, and thus never considered emotion as a potential input for a computer interface (McNeese 2003). That is slowly changing as the online environment increasingly presents itself as a social environment that requires additional authentic inputs.

The development of computers that are able to capture, measure and assess internal physiological signals has been the area with the most progress in this field in recent years. Computers would be able to register a user's emotional state through tiny skin sensor--which the user may have to attach to his body—which can relay physiological information like blood pressure and temperature. Using this data, researchers have been able to detect emotional states like anger, happiness and love (Picard-2000). The proposed uses for this technology are quite varied. Groupware programs that are used in distributing work-loads may be able to use the technology to gather more input from the members of the group to make better decisions. The technology could be implemented in potentially hazardous operations (like driving a car) to alert a user to his emotional state or potentially change the machine's interface to adapt to the emotional state of the user.

This new frontier in computing may have some benefits in education as well, particularly in the constructivist school of thought. Communication could obviously become more rich and contextualized with the additional input of emotion. Discussion and debate via e-mail or instant messaging currently suffer from the lack of context often resulting in misunderstandings. Registering emotions in our online communications could better facilitate interpersonal understanding on many different levels.

More interestingly, the registering of emotion could be used to help students reflect on their personal reactions to topics in their studies. Reflection is a fundamental aspect of learning that is not addressed in many academic settings. Emotion directly affects how people perceive and interpret the world. In order for a student to develop his own knowledge, he needs to understand this process. Developments in this field of affect computing could contribute greatly to this goal.

Conclusion

In the previous section, this paper presented several new programs and research that allow computer learning environments to 'understand' students' needs and thinking processes, and respond to their inputs with relevant information that provokes the student to reflect and/or grow his own understanding about the topic. The two programs that facilitate connectivity—Watson and I2I-- seem to have the most promise in serving to develop constructivist learning through IT. Discovery learning, apprenticeship, and collaborative learning all serve to benefit from these technologies. Disappointingly, Autotutor, like other similar automated tutors, has some shortcomings. The program does well in implementing the idea of proximal development—assisting a student in a knowledge range slightly beyond his abilities-- but the program was only able to understand the student within a certain set of parameters. This confines the student to respond only in a particular way, and prevents the student from introducing other knowledge into his explanations. This is counterintuitive to constructivist beliefs about learning.

This last point demonstrates the disconnect between academics in computer science and those of the learning sciences. Of course, computer programs are limited by the (lack of) progress in the field of artificial intelligence. Perhaps the small, but growing field of affect computing will change their focus. Computers might serve as useful mentors and coaches in the capacity of provoking a student to express and understand his own emotions about a topic. That type of understanding allows students to more fully comprehend how they interpret subject matter and enables them to guide their learning accordingly. This technology is not far out of reach and could contribute significantly to distributing constructivist ideas of learning through IT.

Truly ‘intelligent’ learning systems remains elusive in IT, though there are some ingenious developments in information retrieval and sharing that have significant promise in education. It arguably might be best, if we leave the ‘intelligence’ aspect of learning to traditional human teachers, and leave information and communication management to computers.

References:

Brna & Cox. "Adding intelligence to a learning environment: learner-centered design?" *Journal of Computer Assisted Learning*, (1998), vol. 14, pp. 268-277.

Budzik, Bradshaw, Xiaobin Fu & Hammond, "Supporting on-line resource discovery in the context of ongoing tasks with proactive software assistants." *International Journal of Human-Computer Studies*, (2002), vol. 56. pp47-74.

Cumming, G. "Artificial intelligence in education: an exploration." *Journal of Computer Assisted Learning*, (1998), vol. 14, pp. 251-259.

Graesser, Vanlehn, Rose, Jordan & Harter, "Intelligent Tutoring Systems with Conversational Dialogue." *AI Magazine*, (Winter 2001), vol. 22, n.4.

Graesser, Xiangen Hu, Susarla, Harter, Person, Louwerse & Olde. "AutoTutor: An Intelligent Tutor and Conversational Tutoring Scaffold": http://www-2.cs.cmu.edu/~aleven/AIED2001WS/Graesser.pdf

Hartley, J.R. "CAL and AI—a time for rapprochement?" *Journal of Computer Assisted Learning*, (1998), vol. 14, pp. 249-250.

Jonassen, D.J., (1998). Theoretical Foundations for Educational Research in Communications and Technology, Simon and Schuster MacMillan, NY "Chapter 7: Constructivism: Implications for the Design and Delivery of Instruction," Thomas M. Duffy and Donald J. Cunningham.

McNeese, Michael. "New Visions of human-computer interaction: making affect compute." *Journal of Human-Computer Studies* (2003) vol. 59, pp. 33-53.

Person, Bautista, Kreuz, Graesser, and the Tutoring Research Group. "The Dialog Advancer Network: A Conversation Manager for AutoTutor,": http://www.cogs.susx.ac.uk/users/bend/its2000/person.pdf.

Picard, R.W. "Toward computers that recognize and respond to user emotion." *IBM Systems Journal* (2000), vol. 39, nos. 3&4.

Pilkington, R. "Dialogues in support of qualitative reasoning." *Journal of Computer Assisted Learning*, (1998), vol. 14, pp. 308-320.

This paper is written by Damien Brockmann for the course EDC385G Multimedia Authoring at the University of Texas- Austin.