Mixing the Initiative in Programming through Conversational Authoring

Jeremy Baer
Department of Computer Science and Engineering
University of Washington, Box 352350
Seattle, WA 98195-2350
jbaer@cs.washington.edu

Abstract

We describe a new mixed-initiative programming paradigm called "Conversational Authoring". Traditionally, most programming and authoring environments have been viewed largely as passive tools. The programmer/author provides the initiative when interacting with the system. In conversational authoring, however, the authoring process may be seen as a mixed-initiative collaboration between the author and the system. The system guides the author by asking questions and requesting information, and participates in the construction of the authored artifact. In this way, the burden on the author is lessened. We have applied the ideas of conversational authoring to the domain of authoring pedagogical agents for interactive learning environments, and we are also currently working to apply it in the domain of children's programming systems.

Although many programming systems may be considered conversational by virtue of their turn-taking style of interaction, we are interested in systems that bring a level of "developer-like" knowledge and initiative to the conversation. Application wizards abound in modern programming and authoring systems, and these wizards embed knowledge of a particular task and provide some initiative once the user has started the process. In that way, they provide some of what we are looking for in a conversational authoring system. However, most wizards tend to be short-lived and focus on particular pieces of an authoring task. Additionally, most generate artifacts that cannot subsequently be directly edited by going back into the wizard, but must be edited with other tools. Conversational authoring systems not only contribute knowledge and initiative, but support long-lived interactions that span the entire scope of the author's work with the system in the course of creating a project.

Our model of conversational authoring requires a minimum of two participants. We consider each participant to hold one of two basic roles – author or facilitator. The role of an author is to supply information and creative intent. The role of the facilitator is to elicit information and creative intent from the author and to assist in the process of realizing the author's desired creation.

While conversational authoring systems do not have to be specialized for creating particular types of software artifacts, we can leverage any specialization that is imposed to increase the expertise of the software facilitator. This can be important because the knowledge brought to bear by the facilitator is one of the main benefits of conversational authoring, so the more knowledge we can imbue our facilitator with, the better. Since our computer facilitators embody a certain amount of expert knowledge (even if that knowledge is simply a design methodology), part of their function must also be as tutors.

We define a computer-based conversational authoring system to consist of the following components:

- A set of classes instantiated and configured by the system on behalf of the author.
- A runtime system.
- A Conversational Transition Network (CTN)

The first two items in this definition could be applicable to many different types of authoring systems. However, it is the Conversational Transition Network that brings the "conversation" into conversational authoring. The CTN is the framework that guides the conversation and specifies the computations to construct the authored artifact. It also embodies the programming methodology of the system. A CTN is a type of state machine, consisting of states representing what we call "question points", as well as transitional arcs between these states. At every question point, the system asks the author some sort of "question" and awaits a response. Question points may contain computation, with the exact form of the question being dependent upon the result of that computation. The arcs in a CTN represent transitions from one question point to another and are taken as the result of the author answering the question at the source question point. Each question point may have one or more out-going transitions. If a question point has more than one out-going transition, the choice of which transition is taken depends on the answer that was given to that question, as well as an optional computation.

Additionally, each transition may have a computational execution component associated with it, which is executed whenever that transition is taken. It is these computational steps that allow the system to construct the artifact being authored as the conversation progresses.

The idea for conversational authoring was born out of the need for an authoring tool for pedagogical agents accessible to non-programmers, and this was our first application domain. The field of pedagogical agents draws from previous work done in intelligent tutoring systems, autonomous agents, and educational theory [4]. However, the task of building pedagogical agents is a non-trivial one that falls within the realm of researchers and professional programmers -- not of teachers, curriculum designers, and other non-programmers. Authoring tools designed for non-programmers to create customized content for particular agents are under way, but until now, little attention has been devoted to the question of authoring tools for creating customized agents, given a particular computer-based learning environment. This is an example of an authoring domain in which the methodology to create such an agent is not obvious to the intended authors. Conversational authoring addresses this problem by supplying the programming methodology, which is embodied in the ways in which the system directs the interaction with the author. Within our generic pedagogical agent architecture [1], there are a number of classes of objects that can be instantiated and interconnected to control the agent's behavior. Our conversational authoring system directs the author through the process of creating these objects and does much of the actual instantiation work on the author's behalf, based on the information supplied by the author in the course of the conversation. We have successfully implemented a basic conversational authoring system for our pedagogical agents and integrated it with a constructionist learning environment designed to teach mathematical concepts through digital image processing. This conversational authoring system uses a wizard-like, forms-based interface, in which a large number of the decision making questions are presented to the user in multiple choice form.

We are currently engaged in building several children's programming environments around the model of conversational authoring. We believe conversational authoring is suitable for children because it removes some of the knowledge burden and allows them to create more complex artifacts than they might otherwise be able to. Furthermore, it encourages them to think about abstract components of their projects, because not as much time is spent on the nuts and bolts. The dialogs are in English or nearly English, rather than a programming language, and that may serve to draw in children who would otherwise be intimidated by a programming system. Furthermore, in some systems, we can tie the conversational programming system into a traditional system where the conversational system can not only act as a collaborator but also as a tutor for the underlying system. One such system that we are building is a turtle graphics interpreter in which the user can not only utilize the standard turtle graphics commands such as "FORWARD 100", but can also interact in a more conversational manner to get the system to draw a desired figure without having to know up front how to do it. The system will not only help the user achieve a desired result but also display the corresponding standard turtle graphics commands and provide an explanation of them. We are also building a system that allows children to customize the knowledge and behavior of a generic "chatterbot" conversational agent by having a "teaching" conversation with the agent itself. We are currently working to integrate more general-purpose programming into this system by allowing the user to "teach" the agent how to answer various mathematical questions. Both of these systems employ a natural language interface based on the notions of directed conversations for conversational agents described by Baer and Morgan [2]. Due to the textual and natural language nature of these systems, they are most closely related to children's programming systems such as MOOSE Crossing [3], as well as work on Logo programming tutors [5].

Conversational authoring has the promise of being a powerful paradigm for any number of authoring tasks, end-user programming, and children's programming. Future work remains to be done to stretch its boundaries, define exactly what types of systems it is particularly good for, and to test its efficacy with a wider user base.

References

[1] J. Baer and S. Tanimoto, "A Generic Pedagogical Agent Architecture That Supports Conversational Authoring", *Proceedings of ED-MEDIA 2000, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Montreal, Canada, June 26 - July 1, 2000, pp. 1553-1554.

[2] J. Baer and C. Morgan, "Educational Applications of Conversational Agents", *Proceedings of ED-MEDIA 2001, World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Tampere, Finland, June 25-30, 2001, to appear.

[3] Bruckman, A, *MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids*, Doctoral Dissertation, MIT Media Lab, 1997.

[4] W. L. Johnson, J. Rickel, and J. Lester, "Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments", *International Journal of Artificial Intelligence in Education*, 2000, 11, pp. 47-78.

[5] M. Miller, "A structured planning and debugging environment for elementary programming", *International Journal of Man-Machine Studies*, 1978, 11, pp. 79-95.