ATG Education Research

The Authoring Tools Thread

Jim Spohrer

Introduction

ATG's Education Research Group was comprised of several complementary research threads. In this short paper, I will focus on the authoring tools thread, which along with ACOT, Vivarium, and the Media Lab, was one of the core efforts. ACOT (Apple Classroom of Tomorrow) schools have been a model of effective use of computers in the classroom for over a decade, and ACOT researchers, especially David Dwyer who lead the ACOT effort for many years, were some of the first to identify the critical importance of teacher professional development to promote educational technology adoption. Another well known thread was the Vivarium project lead by Alan Kay, who envisioned ways to empower kids with powerful ideas about computers, new media, and life-long learning. Finally, Apple's Media Lab, led by Kristina Woolsey, was the thread that pioneered the use of multimedia for learning. Part of the strength of ATG's Education Research effort was the way in which these and other research threads wove together.

In this paper I briefly survey some of the projects that were and are part of the authoring tools thread and conclude with some lessons learned. However, before surveying the projects, I will provide a chronological and organizational overview. In 1989, as part of the ATG's Education Research Group, Mark L. Miller formed the Business Learning Research Group (MrFixit, Role'm, and Piaget projects) to focus on simulation-based learning environments and improved intelligent multimedia software development methodologies [1]. In 1991, through a collaboration with Steve Weyer's Intelligent Systems Group (MacFrames project) and Alan Kay's Vivarium Group (MacPal project), I started the End User Environments Group (SK8, Puppeteer, KidSim, Constructo, and Scientists Work-Bench projects) which drew together about a dozen researchers investigating end user programming applied to education and scientific computing. In 1994, our group had grown into the Authoring Tools & Titles Program (many SK8-based projects, including KidSim, Catalyst, MedFly, East/West Authoring Tools projects) building off a DARPA/NSF grant which allowed us to create an R&D collaboration with Apple, publishers, universities. In addition, collaborating with Alan Kay's Learning Concepts Group (Squeak project) a Smalltalk modernization effort was undertaken. In 1996, our research team adopted its current name, the Educational Object Economy Project, which focuses on web-based learning and authoring communities (EOE project), as well as learning platform evolution (WorldBoard project).

Projects

MrFixIt

In collaboration with Boeing, the MrFixIt [1,3,4] project explored ways in which intelligent multimedia technologies could be used to build simulation-based learning environments for maintenance training tasks. The MrFixit prototype included a simulation of a Boeing 737-100's pneumatic system comprised of over 300 components (Figure 1), as well as a rule-based agent that could troubleshoot over three dozen different fault modes of the simulation (Figure 2). I was the technical lead on this project, and first developed the prototype in SuperCard, augmented with a simple rule-based expert system shell I had implemented. Later, MrFixit was ported to SK8.

Role'm

In collaboration with Apple's Field Sales and Support Group, the Role'm [1,2,4] project explored ways in which intelligent

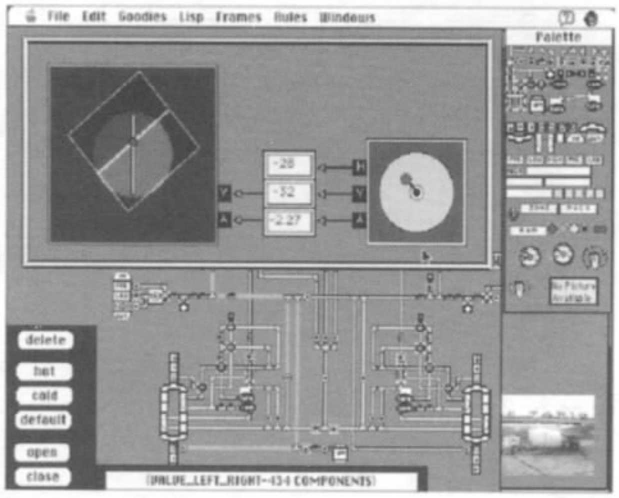


Figure 1: MrFixit, aircraft pneumatic system device simulation

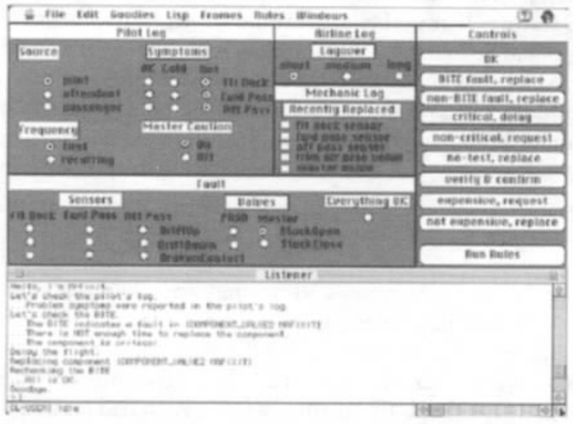


Figure 2: MrFixit, troubleshooting expert system

multimedia technologies could be used to build interpersonal simulators targeted at network solutions sales and technical needs assessment tasks (Figure 3). The Role'm prototype included a menu-based natural language interface that allowed a student to interact with a simulated customer (including a detailed model of the customer's organization and existing computer network). The Role'm prototype embodied Rackman's SPIN sales technique. Arthur James was technical lead on this project, and first developed the prototype in the KEE, Knowledge Engineering Environment. Later, Role'm was ported to SK8.

Piaget

Based on the experiences of building MrFixit, Role'm, and other intelligent multimedia simulation-based learning environments, I formulated an authoring tools strategy and then managed the creation of many dozens of authoring tools. The

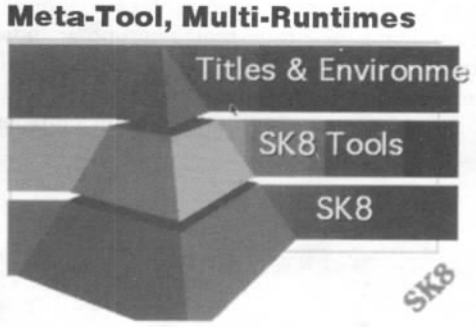


Figure 3: Metatool to create task specific authoring tools

authoring tools strategy came to be known as the 'metatool titlewave (Figure 4)' strategy (or initially, the Piaget project), and was based on the idea that many task-specific authoring tools would be needed to empower non-programming instructional designers and subject matter experts to easily create next generation instructional titles. To create the many task-specific authoring tools, a powerful metatool or tool building technology would be required combining some of the best multimedia features of tools like HyperCard and SuperCard with the best artificial intelligence and modeling technology like KEE. In sum, the authoring tools strategy had three steps: first, create a powerful intelligent multimedia metatool; second, programmers would use the metatool to create task-specific authoring tools; and third, non-programmers would use the task-specific authoring tools to build lots of titles (a 'titlewave' as Chris DiGiano referred to it).

To build confidence in the metatool titlewave strategy, we undertook an experiment: build task-specific authoring tools and then recreate MrFixit with the new tools. Furthermore, we expected that building the task specific authoring tools would not take longer than building the original MrFixit, and using the task specific tools to rebuild MrFixit would be doable by a non-programmer in an order of magnitude less time than building the original MrFixit. We knew that building the original 30 component MrFixit device simulation had taken about one week of SuperCard programming. Plus, the separate components had taken several days each. The intelligent agent that was able to handle three dozen fault cases had required about two weeks of SuperCard programming. In addition, two weeks of debugging an integration were required. In total, we expected a skilled SuperCard programmer would need between one and two months to create a MrFixit-like coached simulation learning environment.

Three task specific authoring tools were needed to rebuild MrFixIt: PiagetDraw (a tool for building the device components, both finite state machines as well as continuous state machines based on pins, grooves, sockets [5,11, 18] and other graphical constraints, see Figure 4), PiagetDiagram (a tool for assembling components into a complete simulation, see Figure 5), and PiagetTree (a tool for building the decision tree logic for troubleshooting the device's failure modes, see Figure 6).

Each of the task specific authoring tools had required about two weeks to design and build, by myself, Dave Vronay, and Alan Peterson. Using these three tools a non-programmer was able to recreate a MrFixIt-like learning environment (with many more components and failure modes modeled) in just under two days (as compared to 1-2 months for a programmer using SuperCard before)!

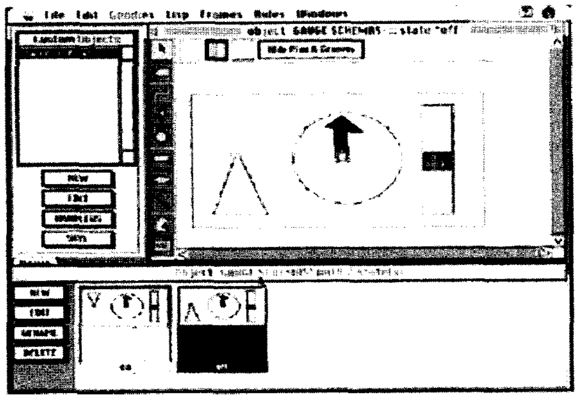


Figure 4: PiagetDraw, to create device components

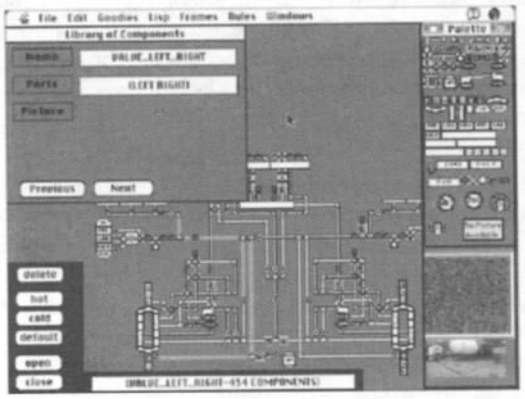


Figure 5: PiagetDiagram, to connect components into a complete device

The benefits of the metatool titlewave strategy were becoming increasingly clear, and so our attention turned to two new problems: creating a suitable metatool and creating a suite of task-specific authoring tools to demonstrate a range of end-user programming techniques. Fortunately, there were already two related projects underway at Apple, MacFrames and MacPal.

MacFrames

Ruben Kleiman in ATG's Intelligent Applications Group had been building an intelligent multimedia application framework in Macintosh Common Lisp. Kleiman called his application framework MacFrames, and we had used MacFrames to build the three prototype Piaget task-specific authoring tools. At its core, MacFrames had a powerful and flexible prototype-instance frame system. Built on the frame system core was an extensive library of prototype media and knowledge objects. However, MacFrames lacked a scripting language and several other features that we believed would be necessary for the metatool to be usable by users familiar with HyperCard. At the time, MacFrames users required Lisp programming knowledge to build the task specific tools. So building on MacFrames, a design for a more suitable metatool was undertaken. The metatool work that grew out of MacFrames came to be known as SK8.

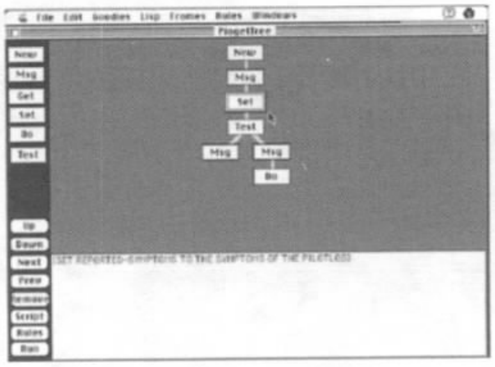


Figure 6: PiagetTree, to create troubleshooting decision trees

MacPal

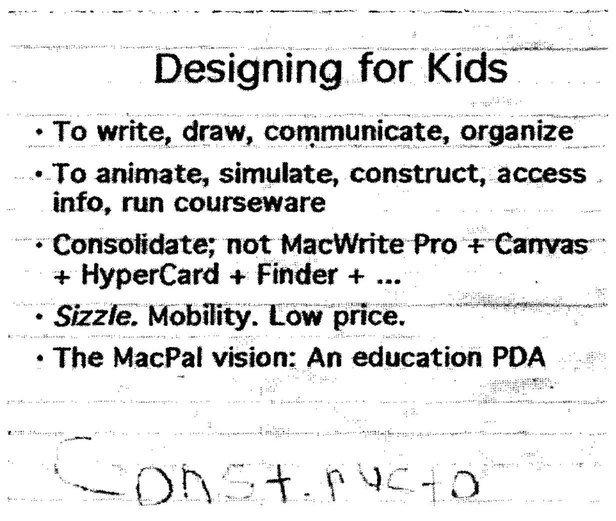
Also at that time, Alan Kay's Vivarium Group had just completed a design, implementation, and test cycle for a kid usable programming environment, called Playground. In preparation for building the next iteration of a kids programming environment and Dynabook-like hardware for it, Alan had created the MacPal working group. The working group attracted over a dozen of the best minds from across Apple who were working in a number of relevant areas, including the area of end-user programming techniques [8]. A guiding principle that Alan Kay instilled in everyone that came to these meetings was the need to embody a set of powerful ideas in the new programming environment that we were all struggling to create. While we wanted to make programming the simple things simple, we wanted the environment to allow for graceful scaling up so that more complex things could and would be created as the programmers became more accomplished in using the environment. A technical report on a variety of end-user programming techniques was produced by the working group. Also, out of the MacPal working group, two key kids programming environment projects emerged: Constructo (general purpose programming) and KidSim (task specific programming).

SK8

SK8, a metatool or tool building technology, can best be described as HyperCard on steroids. SK8 [6, 10, 13, 14] was designed to be appealing simultaneously to HyperCard programmers who wanted a more complete and powerful programming environment to grow into, as well as for novice programmers looking for a first object oriented programming environment. SK8 users program in SK8Script, a natural language like programming environment that unlike HyperTalk has a full prototype-instance object model as well as standard programming data structures conveniently accessible. The graphics model allows any object to contain any other object, so containment becomes a powerful way to build up new objects out of many component objects. In addition, SK8 includes an extensive library of predefined objects that were carefully constructed to make building task specific authoring tools straightforward. For example, connector objects and a port wiring mode make it easy to build authoring tools based on a wiring metaphor; a two dimensional picker object makes spreadsheet and grid-based authoring tools straightforward to construct. Finally, the SK8 Project Builder allows sophisticated projects to be built entirely by direct manipulation. SK8 has hundreds of subtle, but important, productivity features built into it. Over one hundred research project tools were built or first prototypes in SK8 [23].

SK8 is a large programming environment, requiring a minimum of 25MB to operate comfortably. SK8's size is not a problem for professional multimedia developers, since they typically use high end machines with a lot of RAM. However, deployment of large projects is a problem. SK8 projects can be delivered in under 16MB on a Macintosh Common Lisp runtime. In addition, several investigations have explored delivering SK8 projects on Kaleida's ScriptX as well as Java runtimes targeting machines with about 8MB of RAM.

SK8 and its complete set of source code (implemented in Macintosh Common Lisp) can be freely downloaded at http://sk8.research.apple.com/ [19]. Ruben Kleiman was the chief architect and implementor of SK8, though many others including Brian Roddy, Hernan Epelman-Wang, Sidney Markowitz, Chris Flick, Ken Dickey, Philip McBride, John Ulrich, Michael Evins, and others too numerous to mention made contributions large and small.


Designing for Kids

- To write, draw, communicate, organize
- To animate, simulate, construct, access info, run courseware
- Consolidate; not MacWrite Pro + Canvas + HyperCard + Finder + ...
- *Sizzle.* Mobility. Low price.
- The MacPal vision: An education PDA

Constructo

Figure 7: Constructo, overview slide

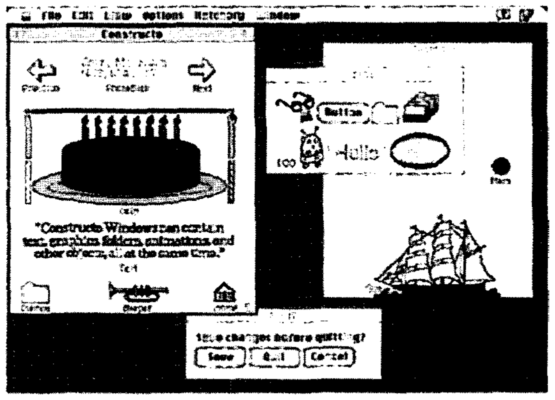


Figure 8: Constructo, screenshot

Constructo

Constructo was an off shoot of Alan Kay's MacPal working group. Constructo was designed to be a programming environment for kids that would introduce them to a dozen powerful ideas in programming and interactivity, including objects, containment, direct manipulation, variables, conditionals, iteration, subroutines, and recursion (Figure 7 and Figure 8). Constructo was noteworthy as an elegant unification of many of the best features of the Macintosh Desktop Metaphor, HyperCard, MacWrite, MacDraw, and MacPaint. Constructo also borrowed ideas from the DiSessa's Boxer system for children programmers. One powerful feature of Constructo was the notion of object halos, that would pop-up around an object allowing a user to easily and directly interact with the objects in many useful ways. Constructo made easy tasks easy to do, and difficult tasks possible because Smalltalk programming was accessible just under the hood. Constructo was implemented in Smalltalk by Scott Wallace and Jerry Morrison.

KidSim (a.k.a Cocoa)

KidSim [6, 7, 10, 17, 20] was another off shoot of Alan Kay's MacPal working group. KidSim was designed to be programming environment for kids that would introduce children to the wonders of programming by empowering them to create simulated worlds. Unlike Constructo, which was designed to be a general purpose programming environment, KidSim was designed to be a special purpose or task specific authoring tool for kids to create simulated worlds and interactive games. Another early inspiration for KidSim came from the teachers who helped co-design the early prototypes in part to meet their need to create simulated worlds similar to the one described in the book "Planiverse" by A.K. Dewdney (Figure 9).

To make KidSim accessible and fun for children, KidSim was designed around a gameboard metaphor. Game pieces could be created, and then placed on a gameboard (a two dimensional

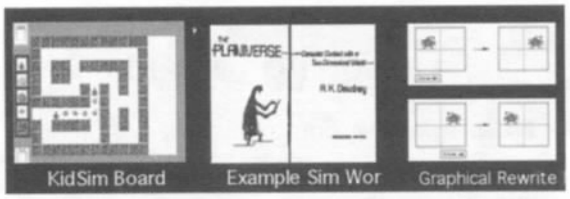


Figure 9: KidSim allowed teachers to build Planiverse type worlds

grid). Pieces could be positioned in such a way that if an interaction between the pieces should occur (such as putting a fish near a shark, so the shark could eat the fish), then a rule defining the interaction could be easily demonstrated via direct manipulation. KidSim used graphical rewrite rules to capture the behaviors of pieces in particular contexts. Graphical rewrite rules are also part of Alex Repenning's Agentsheets programming environment. Once game pieces are defined, rules of interaction are defined, and an initial configuration of pieces is placed on the gameboard, a clock is started and the pieces begin moving and interacting. To make debugging straightforward, the clock can be run backwards. Ten year olds were able to begin building KidSim worlds with as little as fifteen minutes of instruction. Some of the early KidSim worlds included an aquarium simulation (Figure 10) and a hang glider simulation (Figure 11).

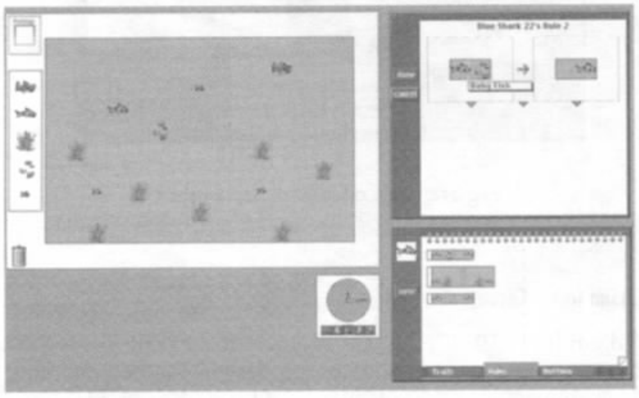


Figure 10: KidSim aquarium simulation

KidSim was created and implemented in SK8 by Dave Smith and Allen Cypher. Later, they worked with Kurt Schmucker and Peter Jensen, who rebuilt KidSim in Prograph. The new product, called Cocoa, is freely downloadable from http://cocoa.apple.com/ [16]. A key feature that was added in the Cocoa implementation was the ability to embed Cocoa worlds in web pages, making it a great tool for kids to build custom, dynamic, fun web pages.

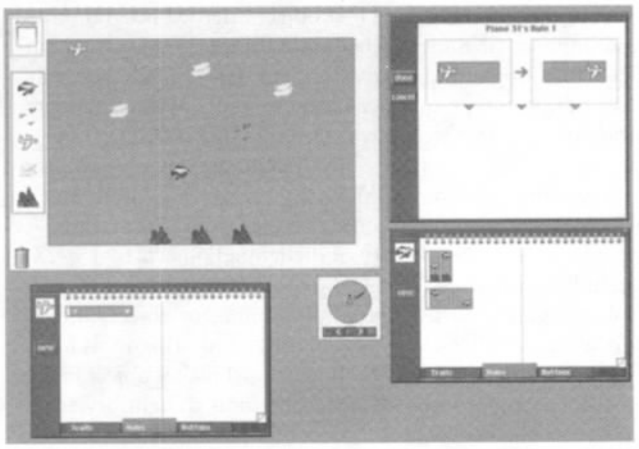


Figure 11: KidSim hang glider simulation

Puppeteer

The Role'm work, aimed at building intelligent multimedia simulators to train people in interpersonal communication skills, had been evolving as well. Enio Ohmaye, had joined our group as a recent Ph.D. from Roger Schank's Institute for Learning Sciences (ILS) at Northwestern University. At ILS, Ohmaye had created a system called Dustin, which used a wiring metaphor to construct sophisticated interpersonal simulators. Ohmaye and James joined forces to explore a new metaphor for authoring interpersonal simulations, a metaphor based on comic strips. The comic strip metaphor became the basis of a task-specific authoring tool known as Puppeteer (Figure 12). To demonstrate the usability of Puppeteer, two advanced high school students were recruited to use Puppeteer to build a second language/second culture interpersonal simulator for business people to prepare for business meetings with Japanese collaborators. The comic strip metaphor proved to be valuable not only during authoring, but also proved to be valuable during maintenance operations to extend existing instructional simulation titles. A valuable component of the Puppeteer project was a grammar authoring tool (Figure 13) used for constructing the form of the allowed natural language interactions with the interpersonal simulations. In addition, MacHeadroom was a task specific authoring tool used in conjunction with Puppeteer. MacHeadroom provided an emotion editor for talking head agents that would move their lips, eyes, and foreheads to convey text in a variety of (color coded) emotional tones (Figure 14). Clusters of synergistic tools such as these Puppeteer tools supported the notion that numerous task specific authoring tools would be used to create a final educational software title or environment.

Scientists' WorkBench (a.k.a. Data Workbench, Commander Data)

The wiring metaphor is a valuable end user programming technique. However, as the complexity of an artifact increases, wires begin crossing, and the spaghetti effect kicks in. Finding a subject domain and a set of users who are comfortable with the wiring metaphor can be challenging. LabView is one tool that effectively uses the wiring metaphor to create virtual

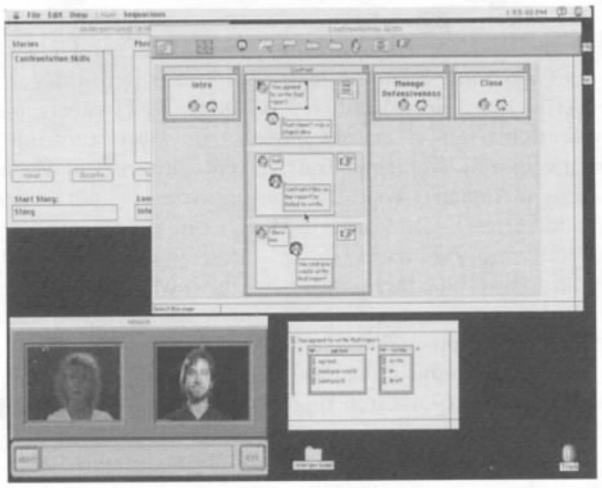


Figure 12: Puppeteer's comic strip metaphor

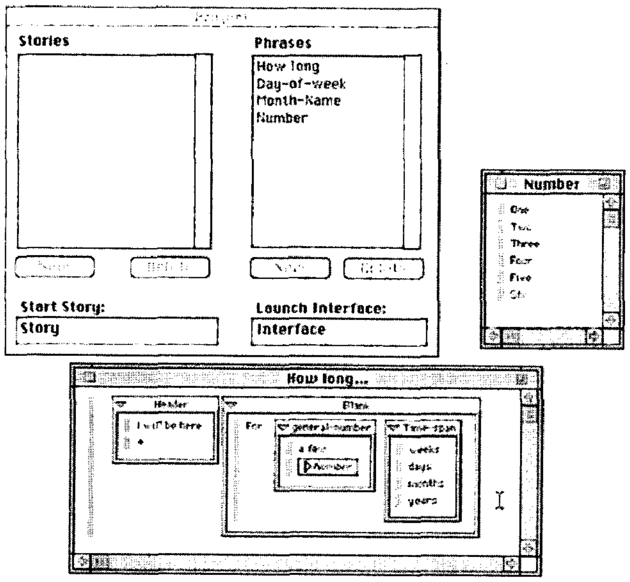


Figure 13: A grammar editor tool for natural language input

instruments. The metaphor is effective in part because the creators of devices are often familiar with wiring hardware components together in real life.

Many scientists we spoke with were comfortable with the wiring metaphor for certain tasks. Rob Wolf and Vince Kirchner created a tool known as the Scientists Workbench that allowed scientists to easily wire together signal and data processing components to data banks and visualization tools (Figure 15 and Figure 16). The Scientists' Workbench was a combined authoring tool and productivity tool for scientists. The tool was interesting from a social perspective as well, as certain scientist became creators of components to be wired together, while others became users of the components.

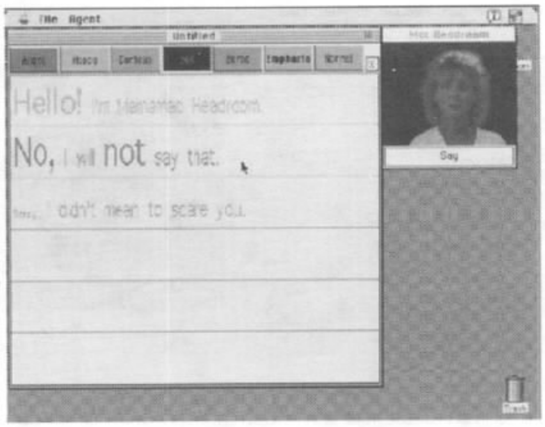


Figure 14: MacHeadroom creates agents that display emotions

Catalyst (a.k.a. Media BluePrint, QuickStory)

Catalyst was the first project undertaken in our group that tackled the challenge of supporting authoring teams as opposed to supporting individual authors. While many of the tools that our group was creating explored end user programming technologies and ways of finding familiar representations for users to express their intentions, models of the world, and knowledge, Catalyst explored the ways groups of people work together to rapidly and creatively produce multimedia software titles. Issues like rapid prototyping, plasticity in design, incremental refinement, managing complexity, sharing workspaces, project management, as well as multiple roles, responsibilities, and representations for information were central to the notion of team authoring. Stephanie Houde, Gary Young, and Lori Leahy designed and implemented a Catalyst prototype in SK8.

MedFly

When Don Norman stepped in to direct ATG, his vision of the future included a multitude of task-specific information appliances. At his suggestion as well as Gary Starkweather's, we began an investigation into task specific authoring tools for information appliance type devices. One of these efforts was the MedFly project.

The MedFly project explored form-based end user programming techniques to create data entry, access, and processing applications for hand held mobile devices, such as Personal Digital Assistants (PDAs) like the Newton. The application domains were medicine and aviation, hence the name, MedFly, coined by Beverly Kane, M.D., the medical member of the Medfly team. Chris Burmester created a SK8-based task-specific authoring tool that allowed form-based applications to be easily created on a Macintosh, and then downloaded to run on Newtons.

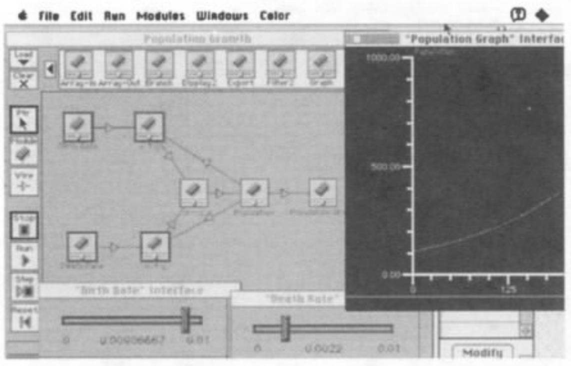


Figure 15: Scientists' Workbench Population Simulation

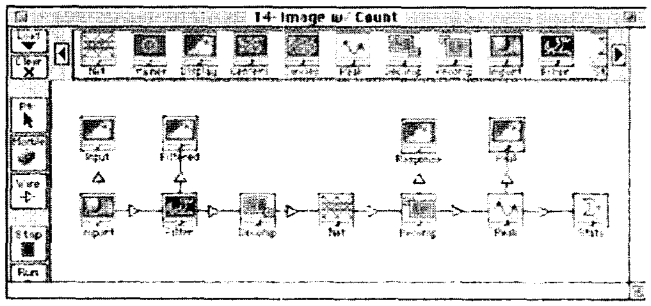


Figure 16: Scientists' Workbench Signal Processing

East/West Authoring Tools Projects

The East/West Authoring Tools Group [12] is an NSF funded consortium composed of Apple, publishers, universities, and government collaborators working together to create next generation authoring tools. Our starting premise is: there is no single best learning environment for all situations, and there is no single best authoring tool. Task specific authoring tools for a wide variety of learning architectures are needed. To be effective the authoring tools must result in order of magnitude reductions in authoring time and be usable by non-programmers. In addition, the resulting learning environments must prove to be engaging, effective, and viable.

During Phase I of the collaboration, authoring tools for next generation learning systems that had been developed at the universities were ported to SK8 under the supervision of Jamie Dinkelacker at Apple. By porting all of the existing tools to SK8, we hoped to better understand the challenges and opportunities for sharing component software for task specific authoring tools, and leverage each others efforts to create hybrid, pedagogically diverse learning environments. Because SK8 and ScriptX were not fully productized (though both are available as technology releases), in Phase 2 of the project the implementation effort shifted to Java and work focused on the creation of an Educational Object Economy (see below).

The East/West Authoring Tools collaboration spans a wide variety of learning environments: construction kits and high functionality design environments (Gerhard Fischer, University of Colorado), including Agentsheets (Alex Repenning, University of Colorado) and HyperGami (Mike Eisenberg and Ann Nishioka, University of Colorado), intelligent tutoring systems (Beverly Woolf and Tom Murray, University of Massachusetts at Amherst), cognitive tutoring systems (John Anderson and Steven Ritter, Carnegie Mellon University), and interactive medical images (Parvati Dev, Stanford University). In addition to those mentioned here, the collaboration spanned many other organizations, many approaches to learning, and many approaches to authoring. More about these projects can be found at the East/West Authoring Tools Group web site: http://trp.research.apple.com/TRP/projects.html

A great diversity of end-user programming techniques have been explored in E/W Tools. For example, while a summer intern at Apple, Stephen Blessings developed a novel combination of direct manipulation, programming by demonstration, and form-based techniques to create production rules for a simple cognitive tutor to teach subtraction with borrowing. He went on to create a Newton version of the resulting subtraction tutor. Over one hundred prototypes of task specific authoring tools were created in SK8.

Squeak

The strategy of creating lots of task-specific authoring tools using a single metatool is not without its problems. As the range of task-specific authoring tools expanded and new 'must have' media technologies sprang up, SK8 was continuing to grow in size and complexity. Alan Kay suggested that a more minimalist approach should also be explored, along the lines of the original Smalltalk effort. In response to this suggestion, Dan Ingalls, John Maloney, Ted Kaehler, and Scott Wallace undertook the task of creating Squeak (named for the small sound of a small animal). Squeak is a modern version of Smalltalk 80, using updated technologies and driven by a minimalist philosophy.

Testament to the success of the Squeak simplification and modernization effort was the fact that after it was released to the Web (with sources) it was successfully ported to four other platforms in just five weeks by self selected Smalltalk experts. Squeak (with sources) is available at: http://www.research.apple.com/research/proj/Learning_Concepts/squeak/intro.html [15]. Today, Alan Kay and the Squeak development team are part of Disney's Imagineering group, and a thriving Squeak community has emerged in support of a simple, open authoring environment. Small is truly beautiful.

EOE

No matter how easy to use authoring tools become, if authors can find what they want (already built) they can save themselves a lot of time. For Java educational applets, the Educational Object Economy (EOE at http://trp.research.apple.com/ [21]) is a good place to 'look before you author.' Even if authors don't find exactly what they want, they may be able to find something that is close enough as well as qualified people with relevant experiences to join their authoring team for the duration of the project.

The Educational Object Economy (EOE) [22, 24, 25, 26] is an experiment in mega-collaboration and virtual communities (both authoring communities and learning communities). The EOE not only acts as an amplifier of individual actions to improve the quality and availability of Java educational applets, but also provides a free web-based community framework for other researchers who wish to experiment with creating virtual communities for mega-collaboration. So far the mega-collaboration has resulted in the EOE being the largest directory to freely available educational Java applets in the world.

Our EOE is a community of people working together to improve the quality and availability of web-based learning materials. Apple, NSF, universities, publishers, and many others have created a first exemplar Educational Object Economy (at http://trp.research.apple.com), and we are now helping others start their own EOEs. For example, Darcy Clark at University of Michigan has set up a Materials Science EOE. A key part of an EOE is web site technology that helps empower community members to work together. An EOE web site must allow members to easily gather, share, and add value to web-based materials of interest to the community. The technology required to set up an EOE is relatively straightforward. However, creating a vibrant community that is actively achieving its goals, and reflecting that activity through the web site, is a challenging task. John Lilly, Martin Koning-Bastiaan, and Melissa Jones have implemented the Generic Object Economy (GOE) starter kit in FileMaker 4 Pro, and it is freely available at the EOE web site. In a project like the EOE, business models (incentive structures) will be as much a part of the breakthrough as the supporting technologies. Therefore, the EOE web-site has a business section (headed by Lori Leahy) where relevant business models are shared. In addition, Jeremy Roschelle has been active in promoting component reuse in the EOE, and Byron Henderson has been active in promoting co-operative organizational principles in the EOE.

WorldBoard

One of the biggest problems with computer-supported learning is that it usually requires learners to sit (passively or actively) in front of a computer screen. Learning and authoring in context, away from traditional desktop computers is one of the next research challenges we plan to undertake. We see the movement from desktop to networked to mobile learning platforms as part of a process of 'learning platform evolution.'

Table 1: Learning platform evolution impacts pedagogy, production, and proliferation of learning objects

Platform	Pedagogy	Production	Proliferation
Standalone	Content	Authoring Tools	CD-ROM
Networked	Conversations	Educational Object Economy	WWW
Mobile	Context	Sensors	Wireless WorldBoard

WorldBoard is a proposed planetary augmented reality system for next generation authoring and contextual learning. With new technologies that provide very accurate global positioning, global wireless communication, and wearable displays, we can foresee ways to "put information in its place." Geocoded information objects will seem to be located in the appropriate places for their use. Nature trails will have trees and rocks labeled. Historical societies will provide time capsule views from different points in a community. Looking at the night sky one will see lines connecting the stars and names of the constellations. Imagine being able to write messages and compose multimedia annotations anywhere on the planet and have them persist in that location. Providing information seamlessly integrated into places opens up a wide range of possible production techniques based on experience capture, annotation, and replay. For more about WorldBoard, refer to: http://trp.research.apple.com/events/ISITalk062097/parts/WorldBoard/default.html.

Concluding Remarks: Lessons Learned

The summary above provides a sampling of major efforts in ATG's authoring tools area. Because over a hundred different tools were created over the past eight years, many relevant investigations had to be left out for brevity sake. Based on these experiences, one may ask "what were some of the lessons learned?" Three lessons stand out:

Lesson 1: Users First

Authoring tools are used by people who want to create quality computer-based learning materials. Whenever a project ran into trouble it was because we had taken our eyes off the true wants and needs of real users. Users had to be involved from two perspectives: cognitive fit (for usability) and social fit (for dissemination). Cognitive fit dealt with the usability of the tool. Co-creating a suitable interface metaphor that was familiar to the users was often a key step in developing the tool. However, even a usable tool might not be readily adopted in the community of users without a dissemination strategy that had been co-defined with that community.

Lesson 2: Complexity Kills

Complexity kills for three reasons: keeping up with technologies that are rapidly evolving on multiple fronts is nearly impossible, learnability and hence dissemination efforts suffer, technology transfer efforts from research teams to product teams flounder. Authoring tools that incorporate innovations on multiple fronts often make great demonstrations, but may not be maintainable, adoptable, or productizable. By contrast, successful tools were ones that had a single key innovation that users could readily see the value of and learn to incorporate into their daily practice, especially when the innovation could be fit into an incremental improvement to an existing product.

Lesson 3: Cognitive Fit Is Easier to Attain than Social Fit

Designing and implementing tools with a good cognitive fit for user and task is an exercise in removing gratuitous difficulties and finding familiar, expressive interface metaphors. Because there is an existing way the task is being performed by existing users (authors), it is possible to invent a new tool idea and quickly test the idea with users to see if it in fact makes a task easier. Often users are able to state with confidence whether or not they believe a new way of doing a task will be easier and more efficient based solely on hearing the idea, without having the new tool implemented. However, designing and implementing a dissemination strategy with a good social fit for the user community is far more difficult. A good social fit means users eagerly give up an old tool and adopt a new tool. The time constant in social fit (new product adoption cycle) is typically much longer than the time constant for cognitive fit (duration of an authoring task). New tools that are free to try, easy to learn, similar to existing tools, demonstrate rapid payback, small and fast are likely to spread rapidly in a community of users.

Like most lessons learned, these seem remarkably obvious in hindsight. What was and is still not obvious is how the broader social and economic dimensions of the authoring tools problem can be changed. For instance, without a robust marketplace for reusable component software, the broader authoring community will have to remain satisfied with incremental improvements to the dominant tools supplemented with occasional innovative niche tools that rarely reach critical mass or continue evolving.

Acknowledgments

This work has been supported by Apple Computer and NSF Grant CDA-9408607. Thanks to the many teachers, summer interns, external research colleagues, and visiting scholars who contributed to this work. Thanks to the following executives who supported these efforts: Rick LeFaivre, Larry Tesler, Dave Nagel, Ike Nassi, and Don Norman.

I must confess it made me a bit nostalgic to think about this stimulating research thread; one that I was lucky enough to be responsible for nurturing and managing, especially in light of the stellar collaborators I had the opportunity to work with over the past eight years.

References

[1] Spohrer, J.C. (1990) Integrating multimedia and AI for training: Examples and issues. *Proceedings of the IEEE Systems, Man, and Cybernetics Conference.* Los Angeles, CA.

[2] Spohrer, J.C., James, A., Abbott, C.A., Czora, G.J., Laffey, J., Miller, M.L. (1991) A role playing simulator for needs analysis consultations. *Proceedings of the World Congress on Expert Systems.* Pergamon Press. Orlando, FL.

[3] Spohrer, J.C., Vronay, D., Kleiman, R. (1991) Authoring intelligent multimedia applications: Finding familiar representations for expressing knowledge. *Proceedings of the IEEE Systems, Man, and Cybernetics Conference.* Charlottesville, VA.

[4] James, A., and Spohrer, J.C. (1992) Simulation-based learning systems: Prototypes and experiences. Demonstration. *Proceeding of the ACM/SIGCHI Human Factors in Computing Systems.*May 3-7. Monterey, Ca. pp. 523-524.

[5] Vronay, D. and Spohrer, J.C. (1993) Pins, Grooves, and Sockets: An Interface for Graphical Constraints. Proceedings of INTERCHI '93.

[6] Spohrer, J.C., Cypher, A., James, A., Kleiman, R. Ohmaye, E., Smith, D.C. (1994) How to make 'complex' software customizable. *Proceedings of the IEEE Systems, Man, and Cybernetics Conference.*

[7] Cypher, A., Smith, D.C. Spohrer, J.C. (1994) KidSim: Programming agents without a programming language. *Communications of the ACM,* 37 (7):55-67. ACM Press, New York, NY.

[8] Chelsey, Chipkin, Cypher, Kaehler, Kay, Kleiman, Miller, Mintz, Morrison, Rose, Smith, Spohrer, Vronay, Wallace (1994) *End-User Programming: Discussion of Fifteen Ideals.* Apple Library Research Note #94-13

[9] Spohrer, J.C. (1994) *Mapping Learning,* Apple Library Research Report #94-12.

[10] Spohrer, J.C. (1995) Site Description: Apple Computer's Authoring Tools and Titles R&D Program. *Artificial Intelligence Review,* Kluwer Academic Press, Netherland.

[11] Spohrer, Richards, Vronay, Chipkin, Kleiman, Miller (Sept 12, 1995) *Graphical Interface for Interacting Constrained Actors: Continuous Machines.* Patent Number: 5,450,540. Assignee Apple Computer, Inc.

[12] *East/West Authoring Tools Group* (site launched June 1995): http://trp.research.apple.com/EWIndex.html and http://trp.research.apple.com/TRP/projects.html

[13] Epelman-Wang, H., Markowitz, S., Roddy, B. (1996) "Graphical Containment in Multimedia Authoring," *Proc. 41st. IEEE Computer Society International Conference* (CompCon 96), pp. 300-304, Santa Clara, February 1996.

[14] Roddy, B., Markowitz, S., Epelman-Wang, H. (1996) "User Interfaces for Authoring Systems with Object Stores," *Proc. 41st. IEEE Computer Society International Conference* (CompCon 96), pp. 305-309, Santa Clara, February 1996.

[15] *Squeak* (Technology Release October 1996): http://www.research.apple.com/research/proj/Learning_Concepts/squeak/intro.html

[16] *Cocoa* (Product Release October 1996): http://cocoa.apple.com/

[17] Cypher, Smith, Spohrer (Oct 15, 1996) *Extensible Simulation System and Graphical Programming Method.* Patent Number: 5,566,295. Assignee Apple Computer, Inc.

[18] Spohrer (Feb. 4, 1997) *Graphical Interface for Interacting Constrained Actors: State Machines.* Patent Number: 5,600,774. Assignee Apple Computer, Inc.

[19] *SK8* (Technology Release March 1997): http://sk8.research.apple.com/

[20] Cypher, A., Smith, D.C. Spohrer, J.C. (1997) KidSim: Programming agents without a programming language. In Jeffrey M. Bradshaw (Ed.) *Software Agents,* MIT Press, Cambridge, MA. pp. 165-190.

[21] *Educational Object Economy* (Launch June 1997): http://trp.research.apple.com/ and http://www.eoe.org/

[22] Roschelle, J., Henderson, B. Spohrer, J.C., Lilly, J. (1998) *Towards an Educational Object Economy.* Technos, Winter 1997.

[23] Roddy, B., Epelman-Wang, H. "Interface Issues in Text Based Chat Rooms", *SIGCHI Bulletin,* Feb. 1998.

[24] Spohrer, J.C. (1998) Authoring Tools, Communities, and Contexts. In Landauer and Bellman (Eds.) *Virtual Worlds and Simulation Conference* (VWSIM '98) Simulation Series Vol. 30, No. 2. pp. 87-88.

[25] Spohrer, J.C., Repenning, A., Dev, P. (1998) Educational Object Economy: Authoring Tools for Simulations and On-

Line Communities. In Landauer and Bellman (Eds.) *Virtual Worlds and Simulation Conference* (VWSIM '98) Simulation Series Vol. 30, No. 2. pp. 115-116.

[26] Spohrer, J.C. and Lilly, J. (1998, submitted) Learning Platform Evolution, *Journal of Interactive Media in Education.*

About the Author

Jim Spohrer is an Apple Distinguished Scientist heading up the Educational Object Economy project. He has publications and patents in the areas of learning architectures, authoring tools, empirical studies of programmers, speech recognition, cognitive science, and artificial intelligence. He received his B.S. in Physics from MIT, and Ph.D.in Computer Science from Yale.

Contact Information

Jim Spohrer
Apple Computer, Inc.
MS: 301-3D 1 Infinite Loop
Cupertino, CA 95014, USA

Phone: +1-408-974-1421
Fax: +1-408-974-5222
E-mail: spohrer@apple.com