The Design of Children's Technology

Allison Druin, Editor

Morgan Kaufmann Publishers, Inc.
San Francisco, CA

Chapter TWO

Kids as Informants:

Telling Us What We Didn't Know or Confirming What We Knew Already?

Mike Scaife
School of Cognitive and Computing Sciences
University of Sussex, Brighton, UK

Yvonne Rogers
School of Cognitive and Computing Sciences
University of Sussex, Brighton, UK

2.1 Introduction

Joe: "So the pike comes down here and it says, 'Where's my dinner?'... It grabs this one... 'Yum yum! This is better than going to the fish and chip shop!'"

Alan: "And when everything is finished, it says, 'Only the pike is left'... and the pike dies... and the pike says, 'Where did all the food go?'... then the pike dissolves... or you just see it as bones."

This is a small extract from a prototyping design session where we asked pairs of 9-year-olds to create a software game, using laminated cutouts, to teach some basic concepts in ecology to children younger than themselves (see Figure 2.1). As evidenced by hours of videotape material collected, there is no stopping the likes of Joe and Alan once their imagination is fired up. But what are we to make of it all? Is this interaction anything more than kids being loveable? Should we really be using this kind of material to get insight into how to design better educational software? Moreover, how can a design team use such inputs? What criteria can they use to determine whether what children come up with is both pedagogically sound and fun? And why should such observations be of any value anyway? Aren't kids simply emulating what's already out there (Kafai 1996)? Don't we already know—as designers and experts in educational technology—what we need to know?

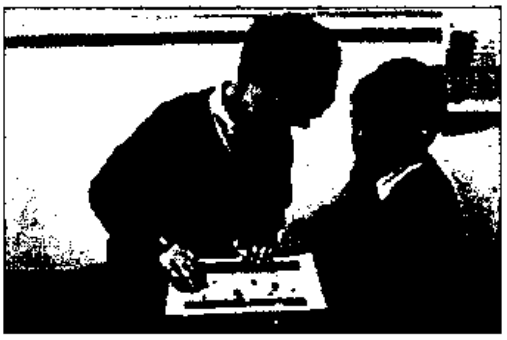
These questions may seem like just another version of a central concern when designing any kind of system or software—when, whether, and how to involve users in the design process and what the actual benefits are of doing so. But, in addition, the involvement of kids, rather than adults, in designing interactive software raises a number of quite specific considerations, and it is these that we shall focus on in our chapter. We shall do this in the context of our own research, the ECOi project.* We have been developing novel interactive software for teaching 9- to 14-year-olds some of the basic concepts of ecology. Our overall research project has several interrelated, complex aims, but we wish to focus here on the insights we gained into the many pros and cons of involving kids within a multidisciplinary design team and how these led to changes in both the design process and the software developed.

User-Centered Design and Adults: Reactive or Participative?

The conventional user-centered design (UCD) approach has been typically to position users as a testing or evaluation service for designers to ensure those users' needs are met (Norman and Draper 1986; Rubinstein and

*ECOi (External Cognition for Designing and Engineering Interactivity), is a two-year project funded by the U.K. ESRC Cognitive Engineering Initiative *(www.cogs.susx. ac.uk/ECOi)*.

Figure 2.1



Joe and Alan, 9-year-olds, doing low-tech prototyping

Hersh 1984). By placing users in this reacting role, designers can obtain a range of feedback as to what is good, bad, and ugly about their designs. However, such a setup means that the kind of feedback obtained from users is primarily based on reaction rather than initiation (Müller, Wildman, and White 1993). A further problem with this kind of asymmetrical relationship is that the onus is entirely on the designers to take on board and translate the users' reactions. Many obstacles can prevent this from happening in a beneficial way, ranging from the designers' own reluctance to reconsider and possibly throw away their own design ideas to organizational constraints that demand the product be shipped before any redesign can be put into practice (Landauer 1996). All too often the actual contribution made by users to the redesign of a system/interface is "too little, too late."

In contrast, the more recently popularized participatory design (PD) approach is to respect users more as partners in the design process and, in so doing, explicitly give them a more equal and responsible role. In this way it is hoped that users can jointly work together with the designers to develop a system that will fit their needs (Schuler and Mamioka 1993). Here "the goal is to provide an equal opportunity design environment in which all participants can contribute as peer co-designers" (Müller, Wildman, and White 1993, p. 64).

User-Centered Design and Kids: Same or Different?

A basic assumption behind the PD approach is that users and designers can view each other as equals. Can this hold when the users are children? In particular, can we have design teams where children are given the same kinds of responsibilities as adults are given in other design team setups? Will designers feel comfortable in taking on board the demands and suggestions of

kids? Or is it more reasonable to position children in a more traditional reactive role, getting them to say what they like or dislike about prototypes that designers create? These questions are relevant for a number of reasons, including the fact that children can't discuss learning goals that they have not yet reached themselves and the strong possibility that interpreting children's dialogue is not as straightforward as may be assumed. In addition, there is the issue of overturning traditional power relations between adult and child, for example, in having children make contributions about the content and the way they should be taught—something that adults have always been responsible for.

It may seem strange to even pose these questions, given the recent spate of reports describing the impressive achievements of involving children in the process of design and evaluation of software packages and programming languages. For example, Druin and Solomon (1996) have pioneered the whole concept of having children as part of a design team, particularly in terms of suggesting metaphors for the designer (see Chapter 3). Kafai (1995) has also employed fourth-graders as software game developers and other, younger children as evaluators of their products in the Game Design Project (see Chapter 6). Likewise, Cypher and Smith (1995) used fifth-graders to test both their own and other children's examples of the rule set in the programming language called KidSim (now called Cocoa; see Chapter 9). More extensively, Oosterholt, Kusano, and de Vries (1996) describe how they successfully involved children throughout the development cycle of a communication device, pointing out how in order to achieve this co-design it is necessary to "enter their world." What is not in doubt, then, is that children *can* be brought into the design process and make a contribution. What is less clear is whether we can generalize about the relationship that they can be expected to have with designers. Put simply, in terms of the two approaches we discussed above, should we view them more as partners or in a more limited role as testers?

The answer to this question, of course, depends on the context and specifics of a given project because there are likely to be different interplays between children and the various adults involved in the design process. Teachers, psychologists, HCI experts, graphic designers, and software engineers will each have their own relationship with each other, needing to interact in different ways. In turn, they will also have different relationships with the children: some will have more direct involvement than others. What we have to do is to ask what might be the optimum kinds of interactions: How can we most effectively involve each person, including the child, in the design process of creating a software product? For example, teachers can often tell us what children find difficult to learn using traditional materials but not what might be effective using the alternative of interactive

multimedia. On the other hand, children are very good at expressing what motivates them in a learning context but perhaps overegg the custard a little and exaggerate when it comes to saying what they find boring. Weighing up and integrating the different contributions is also an important part of the process, and it is unrealistic to take on board everyone's contributions. The design team has to decide how they fit together and whether they fulfill the project's objectives.

Specifying an effective method for involving different people in the design process at different stages is what we have done with our "informant design" framework. Essentially, this involves determining the different phases of design, identifying who will be the informants in these, what their inputs will be, and what methods will be used. Our emphasis is to view different people as informants through our interaction with them. In so doing, it has enabled us as a design team (consisting of an interaction/software designer, an HCI specialist, a developmental psychologist, and an educational technologist) to discover what we did not know rather than try to confirm what we thought we already knew. Such a philosophy is often overlooked by designers following a user-centered design approach in the excitement of demonstrating their own creative designs to users.

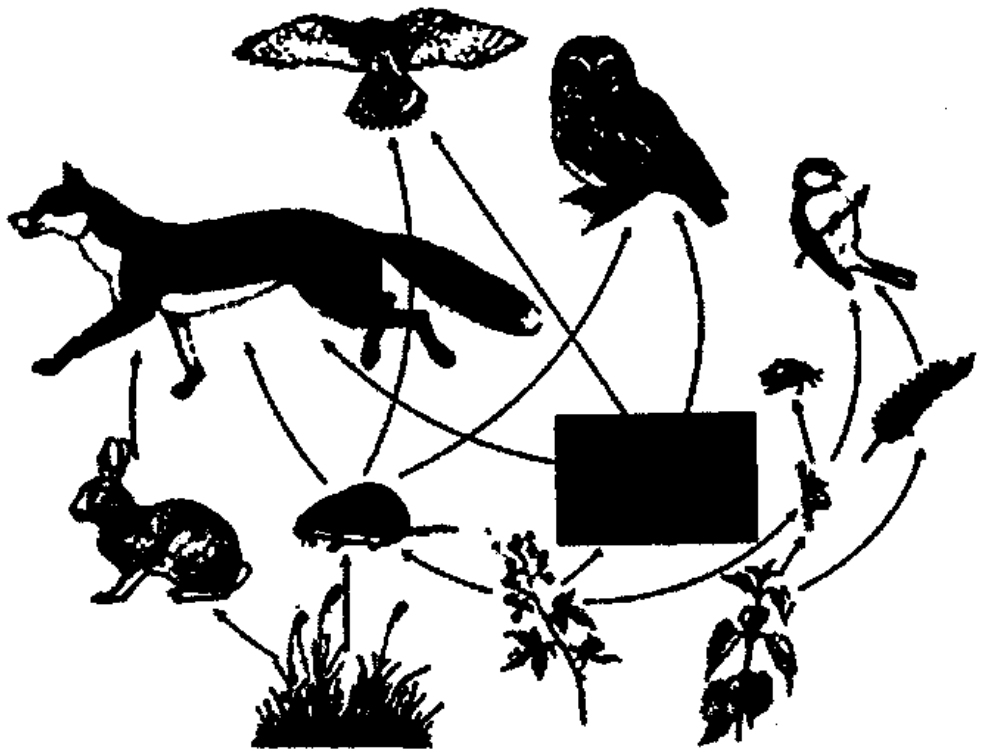
To achieve this understanding, however, it is important to consider the nature of the relationship of the informants with different members of the design team. For example, at the beginning of a project, it is necessary to define the domain and learning problems. In our ECOi project, the educational technologist and psychologist in the design team began by working with teachers from local schools to explicate specific learning goals, to identify the problems with current methods of teaching, and to make a comparison between conventional and interactive media for presenting material. They also interacted with children in their school environment, getting them to evaluate existing materials (e.g., CD-ROMs, textbooks) in that domain to identify what they found to be the main learning difficulties and obstacles to understanding. In parallel, the interaction/software designer created some preliminary sketches and storyboards for the domain space, and the HCI expert operationalized theoretical ideas on interactivity.

In subsequent stages of the project, the various members of the design team worked together in different combinations with the children. Full details of our informant design framework are given elsewhere (Scaife et al. 1997). Here we shall outline how our informant framework arose and describe some of its successes and shortcomings in the ECOi project. We shall draw on our experiences to give a view on problems and successes of our form of child involvement that we hope will be valuable for other projects. Our account here concentrates on the place of the child in the process.

2.2 The ECOi Project—First Steps

ECOi arose from our analysis of the problems that kids roughly 9 years and up seemed to have learning basic concepts in ecology, especially the notion of food webs. On the one hand, they have no problem understanding what a food web is (i.e., that within an ecosystem there is a network of different organisms that eat each other). For example, in a pond, where fish, snails, weeds, tadpoles, and slime cohabit, children readily understand that fish eat tadpoles and that tadpoles eat weeds, but that the tadpoles don't eat the fish and the weeds don't eat the tadpoles. But on the other hand, they are not able to use formalisms, like a food web diagram, to reason about the ecosystem as a whole (see Figure 2.2). Often they get it wrong, redrawing the diagram, showing incorrect relationships between organisms. Typically, they are unable to read information from the diagram about which organisms die as a consequence of one of the other organisms being removed from the

Figure 2.2



A typical food web diagram used in school textbooks to show who eats whom and, thus, the energy flow between organisms. Here one of the organisms has been shaded out in the diagram in order to pose the question, What would happen to the other creatures if the mouse was gone?

ecosystem. The ability to do this kind of inferential reasoning—using the spatial layout to read off the solution by working through the chains of arrows between the organisms—is a fundamental part of understanding ecological concepts (as opposed to simply memorizing a food web diagram, which is typically what most kids are required to do).

This is an example of a far wider problem since there are many other kinds of formalisms, such as cycle diagrams and flowcharts, that are an inherent part of science domains and that have evolved to enable us to make predictions and inferences about the interrelationships between elements and processes. Yet we discovered that, typically, school children are simply not taught diagram-reading skills, with the result that notations such as arrows linking the parts of a food web are not properly understood. Instead, the children rely on their intuitions as to what different symbols mean and how to use them to make inferences, often wrongly since the conventions used in scientific diagrams are counterintuitive to their everyday assumptions. The result is that there is a large conceptual distance between the diagram and the real world, meaning that they are unable to make correct predictions as to what happens when ecosystems are perturbed in various ways.

2.3 The Pedagogical Challenge: Multimedia as a Solution?

We believed that this conceptual gap between the child's everyday experiences and the abstract formalisms that have evolved in science could be closed by using the potential of multimedia software. We argued that explicitly showing the changes that occur in an animated view of an ecosystem in conjunction with showing how those changes are represented in the abstract formalism should enable children to understand better how they could map the two and thus use the formalism to make inferences. In particular, we believed that multimedia software could be designed to allow them to manipulate and construct the food web diagrams themselves and to provide different kinds of feedback—an innovation that simply cannot be achieved using conventional media (e.g., video, books).

To test our ideas, we began by building a simulation of a simple familiar ecosystem, a pond, creating a suite of software modules that showed a variety of diagrammatic interfaces that were dynamically linked to the simulation such that altering one altered the other. For example, removing the token for the tadpoles in the diagram caused removal of the tadpoles in the pond. By this process of "dynalinking," we aimed to allow the child to see the mapping between "real" events and abstract representations of them.

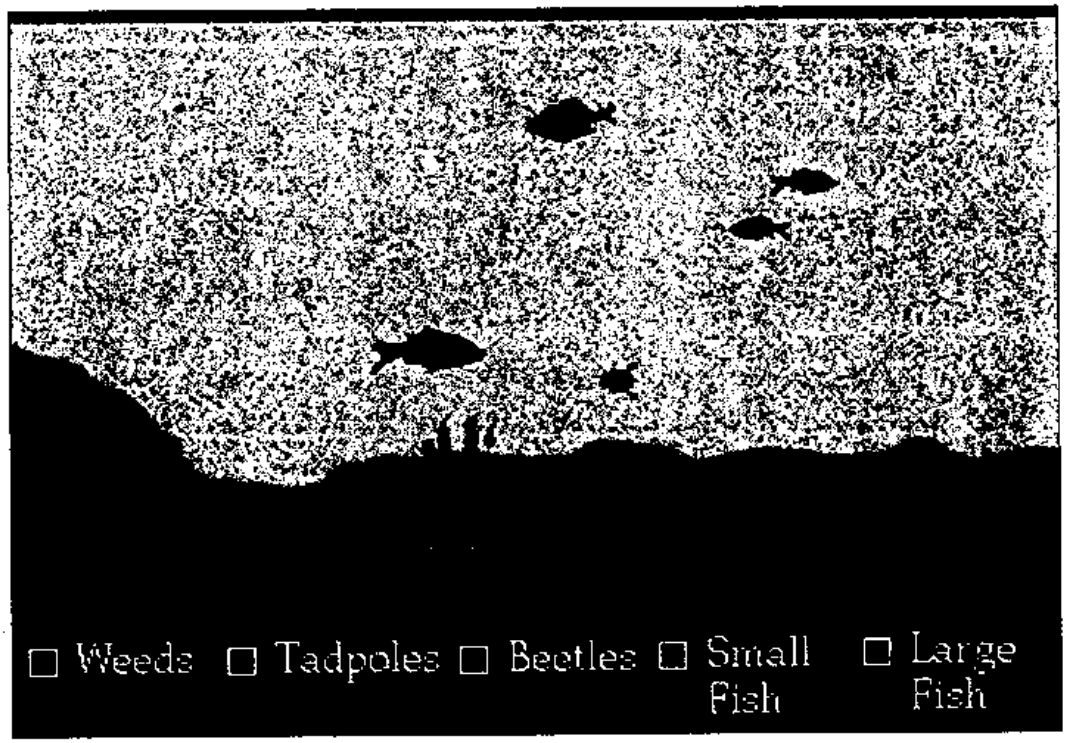
Our First Designs

The initial approach to the design of the simulation was based on an extensive analysis of the problem space and existing attempts to develop learning material. We looked at curricular material identifying relevant ecology concepts and what levels of comprehension were required. We asked 60 children, from 9 to 14 years of age, about their experience of being taught these concepts, identifying issues and teaching methods they found difficult. We also asked 12 teachers of this age range how they would approach the teaching task, such as food webs, and what conceptual difficulties they typically encountered in class. In addition, we got teachers and children alike to evaluate state-of-the-art commercial CD-ROMs that had been developed to teach ecology (Aldrich, Rogers, and Scaife 1998) in order to get a better understanding of what did and didn't work with existing interactive multimedia.

Together, these data served as inputs for the second phase of the design, building a series of prototypes. In this respect, the input of the children and teachers was very important in terms of clarifying for us what was the problem space and why existing learning methods were not working. We also performed a more general developmental analysis of the cognitive difficulties likely to be posed by different kinds of representation in learning. For example: What kind of diagram could we reasonably expect a 9-year-old to understand? This discussion was linked to our analysis of how any external representation was effective, what the pros and cons of different forms might be (e.g., text versus graphic, diagrams versus animation), and how best to combine them (Scaife and Rogers 1996; Rogers and Scaife 1998).

Thus, even at a preliminary stage, the software designer had a basis to begin sketching out a range of ideas for alternative representations. We then decided on the elements of the ecosystem—what we would put into the pond. These had to be simple and familiar enough to be readily understood. Working from our analysis of how interactivity might support learning, we aimed to provide a simple form of diagrammatic linkage to the pond simulation. Using Macromedia Director, our software designer built a simple pond animation, where a number of organisms were visualized (e.g., tadpoles and fish swimming, pond weed moving). In addition, a series of "radio buttons" were displayed below the animation, each labeled as a type of organism. Clicking on a radio button caused the "death" of that type of organism, the exemplars vanishing from the animated pond (see Figure 2.3). However, the action also had a domino effect, resulting in ecologically appropriate removal of other organisms, so that, for example, if the small fish were removed, the big fish—their predators—would also die. A voice-over provided a simultaneous commentary to the actions (e.g., "The small fish

Figure 2.3



The first pondworld prototype

have died . . . now the big fish have died."). In this way the children would be able to see an ecosystem in action.

To determine if our ideas about dynalinking the abstract representation (radio buttons as tokens for species) with the concrete animation were supported, we took the prototype to schools and tried it on some willing 9- to 10-year-olds. The results were not as we had hoped. Although the children had no difficulties in interacting with the animation, their overall verdict of the prototype was harsh: "this is boring" was a representative view. In addition, they often failed to notice the radio buttons and needed to be instructed to click on them. When asked what the relation was between these and the animation, they had a hard time explaining it. This was a setback, but it did not deter us from exploring other ways of visualizing our idea of dynamic linking between abstract and concrete representations. So we went back to the drawing board and sketched out a number of other ways in which we could achieve this form of dynamic interactivity more effectively. In parallel, we discussed why they considered our first prototype to be boring. We realized that there were two immediate considerations. First, did it matter that the

children found the prototype dull? Of course it did. As we all know, motivation is a prerequisite for successful learning. The more difficult second question was, How do we go about fixing this while still sticking to our pedagogical goals?

What we realized was that we were just not on the right wavelength in terms of what the kids found stimulating. So how could we find out? We decided at this stage that it might be fruitful to involve children to inform us in general of what behaviors, features, special effects, and challenges (e.g., games) they enjoyed interacting with at the computer interface. Our idea was that we could usefully employ some of their ideas by integrating them with our own evolving ones about dynamic linking of abstract and concrete representations. Hence, we considered it very important to keep sight of our own goals and not simply hand the design over to the children.

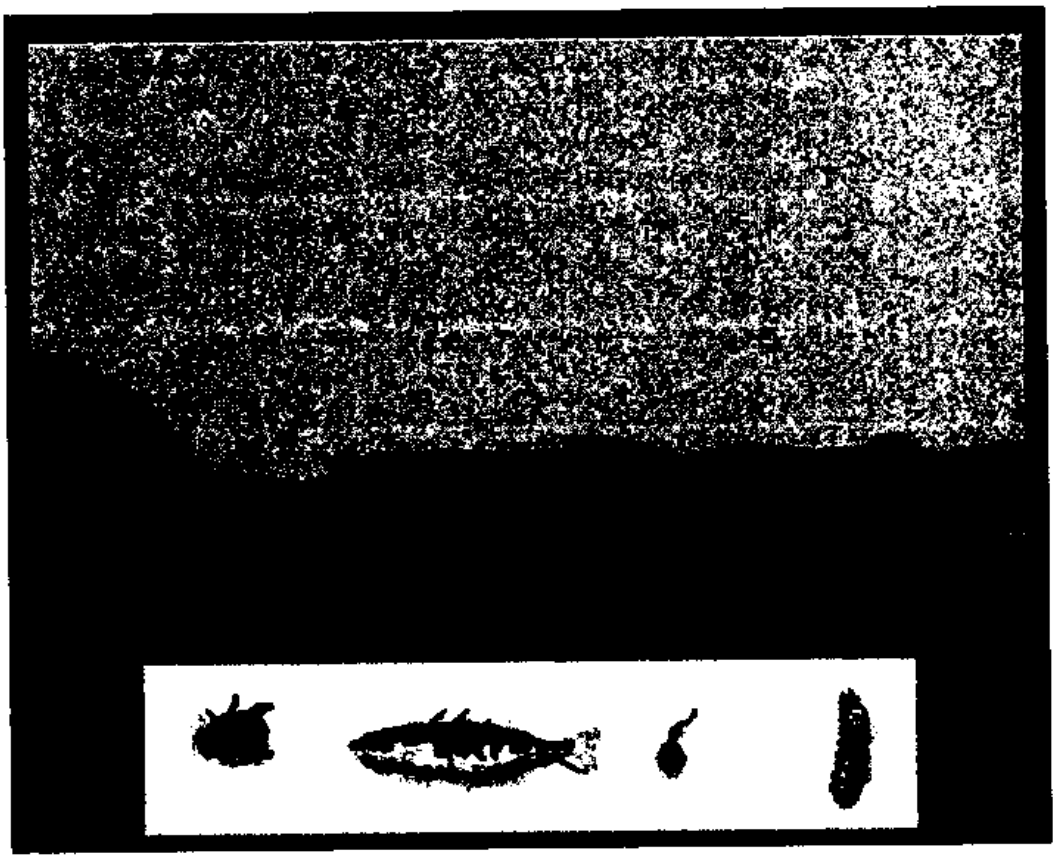
2.4 Low-Tech Prototyping with the Kids

So how did we involve the kids? Clearly, it was totally unfeasible to expect kids to develop software prototypes themselves. Instead, we used a range of low-tech techniques to get the children to imagine what they thought software could be like to teach kids about food webs. One method involved having them make drawings of what the interface might look like and how it might behave. However, this was of limited value, not least because children of this age are in the "literal phase" of drawing (Gardner 1982), spending inordinate amounts of time doing fine details of the creatures and paying little attention to how they would interact and behave. Similar informal observations have been found with many of the Simworld games, where kids enjoy spending much more time creating creatures rather than manipulating their behaviors. Hence, a more productive route turned out to be providing them with already laminated cutouts of the organisms, which the kids could readily manipulate against a background like that of an empty pond (see Figure 2.4; see also Color Plate). In this way, they focused more on the behavior of the ecosystem rather than simply on what the organisms should look like.

A Typical Session

In one session, we asked eight pairs of 9- to 11-year-old children what they thought would make a good CD-ROM for teaching about food webs, encouraging them to use the low-tech materials. We asked them to imagine designing something for children younger than they were, by a year or two;

Figure 2.4



Pond background and creature cutouts used in low-tech prototyping sessions (cutouts not to scale)

they were able to remember their own experiences at this point. Typically they would talk for anywhere between 10 and 20 minutes, coming up with different scenarios for a game or a quiz, which we recorded on audio- or videotape. If they seemed to be stuck, we would ask about the consequences of the imaginary user behaving in a certain way (e.g., making incorrect links in the food chain). At the end, we asked for suggestions about special effects, such as noises made during eating. Throughout the session, the emphasis was on eliciting as many suggestions for animations and games as possible while minimizing input from us.

Results

Working in pairs turned out to be a highly effective way to engender interaction: the kids sparked off each other, sometimes collaborating to extend

each other's ideas, sometimes competing to come up with the "most fun" scenario. For example, one pair kept interrupting each other and saying things like "No, I've got a better idea . . . how about if we had a race-the-clock game where . . ." The majority of children had no problem in using the paper materials to simulate an interactive game, providing animations, special effects, sounds, and feedback. From our point of view, the results of these low-tech sessions were very insightful. Not only were the kids generally very enthusiastic—in contrast to their muted responses to our own first prototypes—but they also revealed a great degree of sophistication in terms of their understanding of the use and potential of interactive software. We can classify the findings into various categories:

Decentered Designs

First, the kids showed a capacity to talk about scenarios in very decentered terms. (The term "decenter" is used in developmental psychology to refer, among other things, to the ability to take another's point of view.) They were able to make comments about the suitability of possible "designs" for a range of audiences. For example:

- If information was delivered as text: "Younger ones wouldn't be able to understand . . . to read it . . . You could have age levels [to adjust the form information is delivered in]."
- On the possibility of gruesome noises for animals dying: "Depends on what age they are . . . If you had them at home they'd have nightmares . . . If they were quite old you could have some really revolting ones . . . Boys prefer more gruesome noises."
- On individual differences: "You could have this go really quick, but then some people like to think about it . . . You could have 'beat the clock' . . . You could choose if you wanted to beat the clock . . . You could pull the hands round so you could have 10 minutes or half an hour."

Structured Learning Designs

Second, they could think in terms not just of special effects, the forte of this age, but also in terms of the metaorganization and aims of a game. For example:

- "You have all the animals lined up like this [shows a food chain], then they go away [i.e., screen clears] . . . then [voice-over] 'Put them in order . . . Who eats who?'"

- "You have questions at the bottom like for [indicates fish], and it says, 'Who do I like to eat?'. . . to see if you've remembered . . . to see if they've been listening or not."
- "You should tell them [users] at the beginning . . . some pond animals eat more than one animal."

Interesting and Controversial Designs

Finally, there were a host of suggestions about how to make the game more interesting by providing lively feedback. For example:

- "So the stickleback looks for food and sees the beetle [zooms the stickleback down onto the prey] . . . grabs it . . . says, 'yum yum' [makes munching noise]."
- "You press return when you are finished . . . [if they've missed a link in the food chain] it says, 'Hey! I eat more than one thing!'"
- "You could have a mouse with three buttons . . . each would have a sticker . . . The left button would start the game up, the right button would finish it, the middle one would save."

One pair also discussed a game where incorrect feeding relationships were permissible, for example, a weed eating a perch. To provide feedback, the children suggested a voice-over narration that would say something like "That's not right, try again." This idea of showing incorrect behaviors that contravened those of an ecosystem became a controversial talking point of the design team. One of the members felt that it might encourage younger kids to believe such behaviors were possible, based on the general assumption that kids are rather gullible and believe what they see on television and other media. The others on the design team disagreed, arguing that, even at such a young age, children have a good knowledge of what is real and what is not, that inanimate objects don't eat live organisms (e.g., weeds do not really eat fish). In the end, we decided to test the latter hypothesis—that providing the wrong feedback in a fun and bizarre way might enable the children to rectify the mistake they had made in trying to sort out the food web and understand why. Thus, in this context, the kids' ideas resonated with some of the conceptual and learning concerns we had ourselves.

Effects on Our Methodology

There were many more suggestions than these. They had two immediate effects on our methodology. First, we were made aware of a valuable resource for design ideas and decided to adopt the procedure in a systematic way.

Second, we realized that we had been misreading the role of the child in the process of developing educational software. Our position had been that we knew what was appropriate for the animation, both in terms of content and interface design. We had seen the children's interactions with the prototypes as doing useful work—validating cognitive assumptions about the comprehensibility of the animation, for example. But we had failed to take on board the extent of their expectations and understanding of both the failings of much of the software they had already encountered (in commercial educational CD-ROMs) and of the potential the medium had (e.g., extrapolating from their experience of electronic games). The most imaginative of the kids could also map this potential onto their insights about learning problems in the domain.

These revelations led us to reconsider how we viewed the child in our design process. The richness of their knowledge about the genre of interactive software and the coupled ability to project this into scenario design led us to view them as analogous to the "native informant" of anthropology. For us this implied that kids are aware of aspects of the use of technology that we are not sensitive to and that we need to be told of. This perspective also holds for others involved in the design process—for example, teachers with their knowledge of the nuts and bolts of what works with a class of 30 or educational psychologists with their understanding of the fit between modules of the curriculum. The end result of this process was to rework our model of collaboration between members of the design team and the user population. So by "informant design," we intend a method for going between privileged observations from potential users and ourselves with another set of skills.

2.5 The ECOi Project—Further Steps

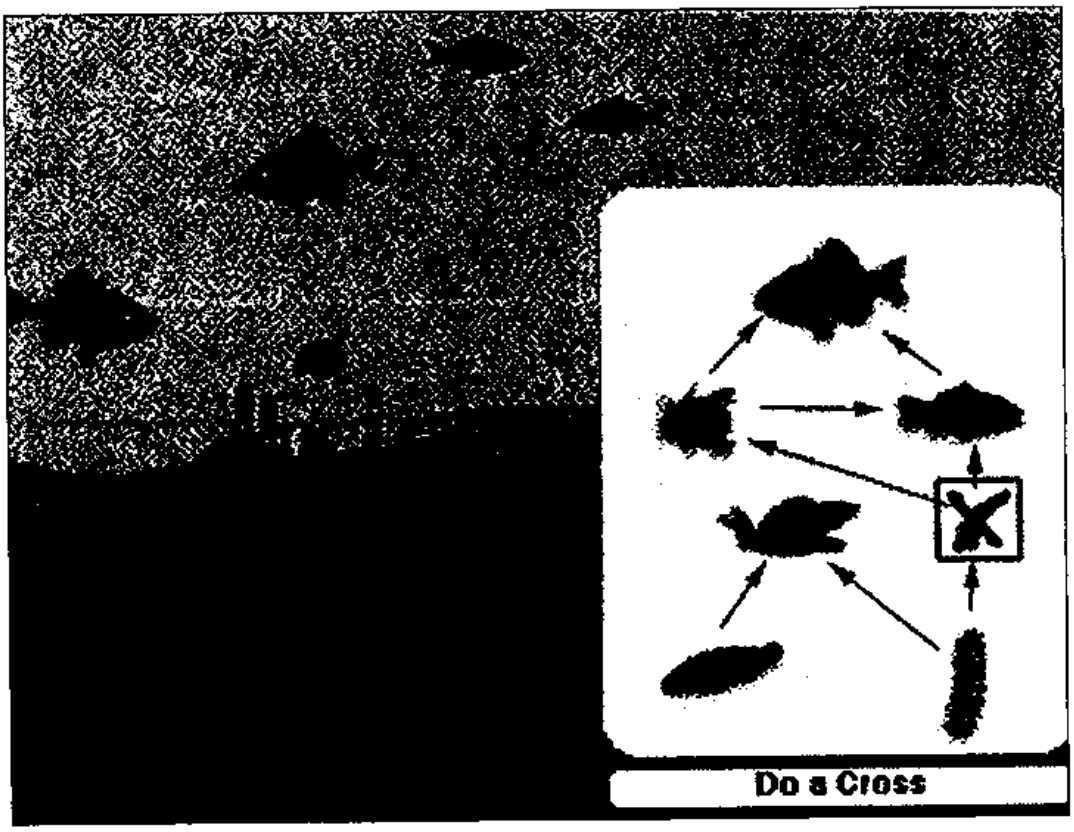
Our interdisciplinary design methodology evolved to incorporate informant design in a number of ways. The most important was to work much more closely with the kids in ways other than having them "just" react to our software prototypes. This was done in two ways.

The first was to systematically use the low-tech prototyping method to develop our suite of software modules, by involving the kids from simple interactions with a pond simulation through to more complex scenarios requiring various manipulations of food web diagrams. The end result of this process was a suite of five modules that the kids worked through in sequence. The first was a simple demonstration of a few creatures in a pond; the last module required them to indicate the effects of deletion of a species

from the ecosystem by crossing out any other species that would die out as a result (see Figure 2.5).

The second was to try to involve the software/graphic designer more directly with the kids, rather than just trying to supply him our own filtered versions of their low-tech designs or responses to his prototypes. Initially we had the designer go out to schools and observe the kids doing low-tech prototyping work. This approach had the merit of giving him a feel for their ideas, which can be lost when viewing these encounters on videotape, as well as a chance to float ideas in conversation with the children. A second liaison was to have the kids come to the design studio and sit down with the designer. The idea here was to have the kids look at prototypes in development and, where possible, have the designer make on-the-fly changes. Of course, there are severe constraints on how many changes can be done there and then—for example, 3D rendering is a time-greedy process. However, the process did work very well for some aspects, especially for special effects.

Figure 2.5



"Eraser-web" module in which children have to delete species in the pond by placing a cross on the food web diagram (part of screen only)

One example of on-the-fly, high-tech prototyping was to get the kids to help us co-design some animations to convey various incorrect behaviors in the ecosystem that would have "kid appeal." We wanted the kids to come up with their own ideas for how this kind of "incorrect" feedback should be presented—as discussed earlier, a controversial learning method, but which had, interestingly, been voiced by the kids themselves in the low-tech prototyping session as something that should be included in a software game. In particular, we wanted them to come up with suggestions of specific details for animations in Pondworld for feeding relationships that did not happen in real life. To do this kind of co-designing with high-tech prototyping, we got pairs of children to work in the studio with the designer. To put them in the picture, he first demonstrated the existing prototype modules to them, explaining the kind of educational software we were developing. In particular, their attention was drawn to the existing animations of correct feeding relationships. They also played with the software prototypes, getting a feel for what could be currently done with them. The designer described to them the incorrect scenario of a weed eating a fish and asked what kinds of things should be included in an animation of this. The kids took immediately to the idea and began discussing how the animation should be quite gruesome, with lots of blood coming out of the fish. They then described what the behavior of the weed should be like and the accompanying sounds. Below is an extract of the conversation that went on between the designer and a pair of 10-year-old girls. It begins at the point where the designer asks the girls what the weed should do:

Child 1: "The weed should get bigger and wrap itself around the fish and pull it down and start eating it." [Child makes lots of gesturing to show how the weed should do this.]

Designer: "OK, and it would make a noise at the same time?"

Child 2: "Yes!"

Designer: "We could do that. What else? Could we do some different sounds at the end to say it's finished eating or something?"

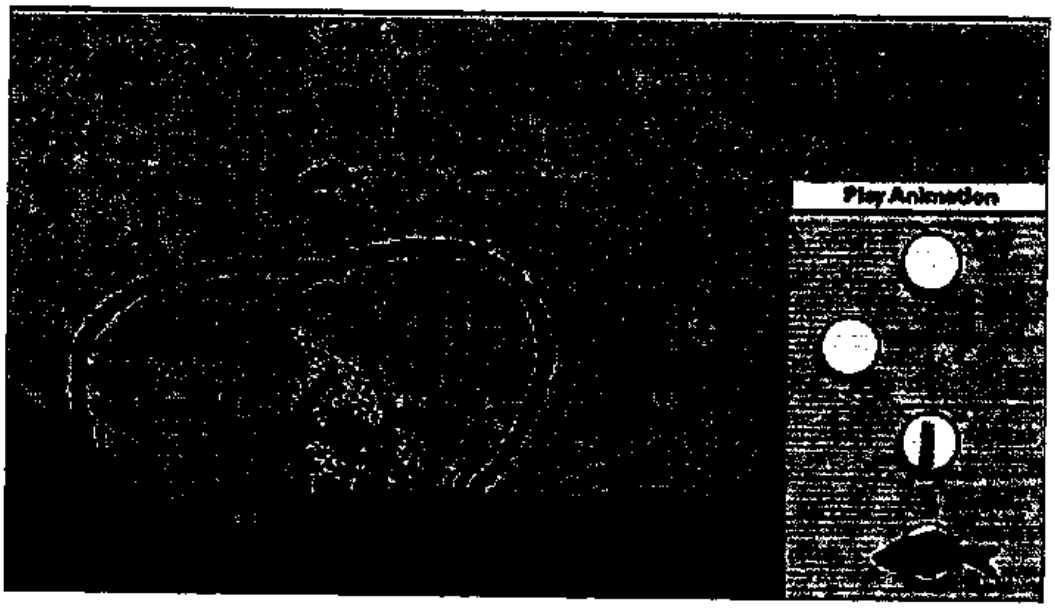
Child 1: "Yes, a burp!"

As can be seen, the designer tries to elicit suggestions from the children and then lets them know if he thinks they are doable. Following this session, the designer set about implementing some of their ideas, including the suggested visual and sound effects (see Figure 2.6; see also Color Plate). This was done using SoundEdit and Director tools. When the kids returned a short while later to the design studio, they watched the new animation in

the making, helping out with some of the sounds. They were impressed by how their ideas had been transformed into actual software. We also got other kids to interact with the redesigned software, and when they happened upon the "bloodbath" animation, they burst out laughing, exclaiming, "That's so funny!"

These on-the-fly, high-tech prototyping sessions showed us that it was possible to get the software designer to work more closely with the kids and to take on board some of their more imaginative and kid-appealing ideas. It also made us aware of the value of using kids to react to existing high-tech prototypes by making suggestions as to how specific components could be developed. A possible problem with this approach, however, is that the designer could be led astray by the multitude of ideas elicited from the kids that end up conflicting with the learning goals specified by the psychologists. For example, it was possible that too much emphasis on the special effects in the incorrect animations might distract the child's attention away from the underlying pedagogical goal of the animations—to get them to confront their own incorrect or incomplete understanding of the food web diagram and, in so doing, modify it. However, this did not happen: instead

Figure 2.6



Snapshot taken from "bloodbath" animation, in which the weed eats the fish. The child has deliberately placed the fish and weed in the wrong places in the food web diagram, resulting in a reversal of normal eating behavior. Chomping sounds recorded from children accompany the animation. The fish also shrinks until it eventually disappears with a loud burp.

the kids tested on such modules appeared to benefit from having both dynalinks and "incorrect" animations. They were kept motivated by the fun elements in the animations while also seeming to understand better the relationships between the organisms in the food web diagram—a desirable balance between the fun factor and the learning factor.

These kinds of involvement with the kids in the design process were just one aspect of a complex set of interactions we coordinated between teachers, psychologists, an HCI expert, and the designer (for more details, see Scaife et al. 1997). There were, of course, other kinds of co-design sessions, where contributions were made by different members of the team in collaboration with the designer. The emphasis in this chapter has been to illustrate how kids can make a valuable contribution when they are treated as informants rather than as partners or simply as testers. So where does this leave us in terms of evaluating the role kids can play? If we stand back from the specifics of our own project goals, we believe that there is some value in giving a brief summary of some of the concerns and practical problems that arise with our way of working.

2.6 Being Selective: Which Method and Which Ideas?

At a recent CHI conference where we presented this work, we were asked whether we weren't guilty of romanticizing the potential role of kids in low-tech prototyping. What have children got that adults haven't? This is an issue with a practical face. On the one hand, the kids come up with many wonderful suggestions that the design team would not have come up with and that, in our case, greatly enhanced the interactive software. On the other hand, many of their ideas are completely unworkable in computational terms and, furthermore, could have conflicted with the pedagogical goals of the software. They may also clash with the established feel of the package developed so far, either in terms of the content or the interface. So how do we know when to say yes and when to say no to kids' ideas?

The problem of selection, from a range of suggestions, is indeed a hard one. It is easy to settle on the ideas that are easy to implement, such as sound effects for dying animals, or to focus on just those ideas that gel with what we previously came up with. The latter approach is another version of the "confirming what we thought" motivation for just having them rate the prototypes for usability. However, there is also a Hobson's choice here. Simply relaying the multitude of ideas elicited from the low-tech prototyping sessions, in an unvarnished way, to the designer can be confusing and

overwhelming—particularly out of context of the original scenario. But translating the set of ideas into an implementable specification means that we need to make principled decisions about the extent to which the suggestions are compatible with the evolving design blueprint.

Having the kids come into the studio also has its problems. We have already mentioned the constraint of what design work can be implemented on the fly. Like many adult users, kids do not have a good sense of what is possible or not. Also, kids will focus more on the fun aspects of the software, tending to suggest how the sounds can be souped up, more blood can be introduced, and so on. In contrast to the freer format of low-tech prototyping, they don't talk much about learning issues. This trade-off resonates with a point discussed by Wong (1992): interfaces that are presented as rough sketches are often much more appropriate prototypes for eliciting responses from users than more finished interfaces because they prevent users getting too fixated on low-level issues, such as what size a button should be, rather than asking more general questions, such as whether buttons are appropriate for the application in hand. In many ways, the inherent flexibility of the low-tech "design your own software" format lends itself to more general suggestions than eliciting responses with the high-tech software prototypes. When confronted with a piece of software that is already designed by any user—be it child or adult—the evaluating users are constrained to make suggestions at a lower level of detail.

In our view, both low-tech and high-tech methods have their place in the design process. The issue is to know which method to use to best effect when involving children during the design process. In our project we used both in tandem, reviewing the outcome in relation to our own pedagogically and HCI-based evolving designs. We should also stress that our design project was very much driven by our original pedagogical concern—to try to get kids to understand better and be able to make inferences from formalisms. Hence, for us, our selection of kids' ideas was largely influenced by pedagogical criteria. Specifically, we wanted to ensure that the resultant software supported our cognitive assumptions about the learning benefits of using dynamic linking of multiple representations at the interface. Many of the specific design ideas coming from the kids were not used, although those that were proved to be highly motivating. (A side effect of this low-tech exercise was that it provided a chance for the kids to reflect on their learning and hence may in itself be an effective learning method.) At a general level, therefore, we can say that kids' ideas are most useful in helping us design the motivating and fun aspects of the educational software—a genre that we as adults are not necessarily tuned into.

2.7 Revisiting Our Informant Design Framework: Kids as Partners or Testers?

Managing an informant design framework involves making just as many compromises as with any other approach. We had to pay careful attention to the inputs not just from kids but also from teachers and educational advisers to maintain the original educational aims of the whole package. In practice, this meant balancing a number of different aspects: learning goals, interface design, fun factors, and technical feasibility. In effect we came, if not full circle, then at least through 270 degrees, as we became more realistic about the contribution children can make to the design process of our interactive software. In terms of the original dichotomy between user-centered design and participatory design, our position is somewhere between the two. We believe in involving the kids as more than reactive critics because they have much to tell us about motivational and genre expectations that we simply don't have good intuitions about. However, we do not treat kids as full partners either, as we are aware of the extent that they can be involved, because of limitations on their knowledge, time, and experience.

2.8 Some General Concerns for a Kid-Centered Informant Design Framework

Our approach, as this book demonstrates, is just one of many. We don't claim its supremacy over others, not least because different contexts will allow different roles for the kids. However, insofar as there are generalizable lessons for us, we have formulated a set of concerns that we believe will apply to projects where children are involved in the design process. Considering at least some of them may help others to plan projects accordingly, to be realistic about how children can contribute within a design team, and not to overromanticize their potential input.

The Selection Problem

Given the imagination of the kids and the endless stream of suggestions, how does the design team make a principled choice?

Comment

It is important to have criteria to determine what to accept and what not to with respect to the goals of the system. This is especially important for

educational software, where interface and fun factors can conflict with learning goals. You need to ask what the trade-offs will be if an idea or set of ideas are implemented in terms of critical "kid" learning factors: that is, how do fun and motivation interact with better understanding? For entertainment software or other technological developments, a different set of criteria and trade-offs will be relevant (e.g., compelling, enjoyable, not boring).

The Focus Problem

Adult assumptions about what is effective may go unnoticed because kids don't necessarily focus on the details of the software that have been designed specifically to support learning goals. How do you deal with this mismatch of expectations?

Comment

Kids may not be sensitive to the learning goals of the software and overlook or use components differently from anticipated. For example, the implementation of dynalinking in one of our prototypes appeared to act more as a mechanical device rather than, as we had hoped, a conceptual learning device. The kids did not pick this up because they were not aware of what it was supposed to be doing. Therefore, involving kids both in the design and evaluation process is important to be able to detect aspects of the software where there are mismatches between expectations.

The Dialogue Problem

Kid talk is not adult talk, and so there can be a translation problem between what they actually say, what they want to say, what we want to hear, and what we actually hear.

Comment

We tend to assume that we can just understand what the kids are getting at, but is this so? We must remember that kids have a different conceptual framework and terminology than adults, and so we need to be aware of the need to speak a common language.

The Individual Problem

How do we design software that caters to the learning needs of the huge variety of kids?

Comment

Clearly, there are big individual differences in what makes learning hard: some kids need more motivation, some get bored very quickly, others need specific educational targets. Furthermore, what works for 7-year-olds may not work for 9-year-olds. So is it possible to design a fully inclusive package for all abilities? Are some formats universally appealing (e.g., quizzes, games)?

The Authoring Problem

Should we just get out of the way and give kids the software tools to do the design?

Comment

There are now software tools geared specifically for kids (e.g., KidPix, HyperStudio). But where should their input end? Again, the extent of their involvement will depend a lot on the kind of product being developed. There may be more scope for involving them as partners for entertainment systems. For educational packages, viewing them as informants is more useful and realistic, particularly when we recall the difficulty of having them define learning goals.

The Single-Method Problem

Is informant design useful for other areas?

Comment

What can kids most usefully help us with (e.g., identifying learning problems, genre expectations)? Is it possible that kid-centered methods can generalize to other areas? For informant design, we realize that it is both cost-effective, targeting expertise in specific areas, but also cost-intensive since not every project will be able to take the time to involve all our informants in the way we did.

9 Conclusions

Our experience has been that the involvement of kids at different stages of software development can clearly bring significant benefits. First, the products are developed in a more efficient way, with the use of low-tech prototyping a highly effective means for meshing kids' design ideas with inputs from other informants. Second, the children's inputs ground our initial approaches to the educational issues, reducing the distance between what we might consider a "good" solution to a learning problem and what they actually find effective and motivating from their own perspective. Finally, we have found the informant design framework to be an insightful and effective approach to design, getting us as a design team to stand back and discover what we did not know, rather than simply trying to confirm what we thought we knew already.

Acknowledgments

This work was supported by the U.K. ESRC Cognitive Engineering Initiative award number L12725103. Many thanks are due to the other members of the ECOi team: Matt Davies, for his outstanding design skills, and Frances Aldrich, for her dedicated field work. We also thank schools in East Sussex, U.K., and Los Angeles, California.

References

Aldrich, F., Rogers, Y., and Scaife, M. 1998. Getting to grips with "interactivity": Helping teachers assess the educational value of CD-ROMs. To appear in *British Journal of Educational Technology.*

Cypher, A., and Smith, D. C. 1995. KidSim: End user programming of simulations. In *Proceedings of CHI '95.* New York: ACM Press, 27–34.

Druin, A., and Solomon, C. 1996. *Designing multimedia environments for children: Computers, creativity, and kids.* New York: John Wiley & Sons.

Gardner, H. 1982. *Art, mind and brain.* New York: Basic Books.

Kafai, Y. B. 1995. *Minds in play: Computer game design as a context for learning.* Hillsdale, NJ: Lawrence Erlbaum.

Kafai, Y. B. 1996. Software by kids for kids. *Communications of the ACM* 39: 38–39.

Landauer, T. 1996. *The trouble with computers.* Boston: MIT Press.

Müller, M. J., Wildman, D. M., and White, E. A. 1993. "Equal opportunity" PD using PICTIVE. *Communications of the ACM* 36(4): 64–65.

Norman, D., and Draper, S. 1986. *User centered system design.* Hillsdale, NJ: Lawrence Erlbaum.

Oosterholt, R., Kusano, M., and de Vries, G. 1996. Interaction design and human factors support, in the development of a personal communicator for children. In *Proceedings of CHI '96.* New York: ACM Press, 450–457.

Rogers, Y., and Scaife, M. 1998. How can interactive multimedia facilitate learning? To appear in *CD-ROM Proceedings of the First International Workshop on Multimodality in Multimedia Interfaces.* Menlo Park, CA: AAAI Publications.

Rubinstein, R., and Hersh, H. 1984. *The human factor: Designing computer systems for people.* Burlington, MA: Digital Press.

Scaife, M., and Rogers, Y. 1996. External cognition: How do graphical representations work? *International Journal of Human-Computer Studies* 45: 185–213.

Scaife, M., Rogers, Y., Aldrich, F., and Davies, M. 1997. Designing for or designing with? Informant design for interactive learning environments. In *Proceedings of CHI '97.* New York: ACM Press, 343–350.

Schuler, D., and Mamioka, A., eds. 1993. Participatory design: Principles and practices. Hillsdale, NJ: Lawrence Erlbaum.

Wong, Y. Y. 1992. Rough and ready prototypes: Lessons from graphic design. In *CHI '92 Conference Companion.* New York: ACM Press, 83–84.