Designing Children’s Software

Abstract

For a long time the fields of Human Computer Interaction (HCI) and educational technology worked independently of each other. Poor usability in software for children reduced the software’s effectiveness. Gradually, the two fields began to merge and the role of children in the design process changed. Children went from complete exclusion from the design process to becoming legitimate members of the design team. There are now conferences dedicated only to the design of software for children.

1 Introduction

Human computer interaction (HCI) is an important research area in computer science. It attempts to discover efficient and convenient ways in which people can interact with computers. When the field emerged as a legitimate area of research, it primarily focused on the interaction between adults and computers. Slowly, as technology became more widespread, researchers realized that traditional methods used to develop systems for adults did not translate seamlessly to developing programs for children.
2 HCI Background

The most standard way of interfacing with computers is to use a screen, keyboard, and a mouse. Using this mode of interaction, graphical user interfaces (GUIs), such as Microsoft Word and Mozilla’s Firefox web browsers, are standard. When creating these types of programs, many goals of the developers pertain to usability. Usability is a general measure of how well users of a program perform. The measure is used to test the success of the program’s interface, not the user. Aspects of usability usually include the following [16]:

- How long it takes for users to figure out how complete a certain set of tasks.
- How many errors (and what types of errors) people make when trying to complete certain tasks.
- How long users remember how to complete certain tasks.
- How much users enjoy using the system/how satisfied users are with the system.

A significant amount of research has been done to find a straightforward process for developing interfaces that are very usable. Most of this research, especially in the early days of HCI, revolves around adults. Coverage of all the variations of this process lies outside the scope of this paper so a more general overview will be given. There are six main stages to successful interface development.[16]

The first stage is to develop product concept. During this stage a number of organizational tasks are carried out. Developers must establish their business objectives, set up a
usability design team, identify the users of the system, set a schedule and budget, and try to identify technical and environmental issues that developers could encounter during the development process. Note that design teams are sometimes comprised of a diverse group of people that will include interface designers and possibly potential users or other representatives from the client. Originally, however, if the software was being designed for children, children would not be included as a part of the design team. In terms of usability, identifying the users of the system is crucial. Gathering information about the age, computer background, educational background, and general attitude of the user group goes a long way toward creating a successful interface.

During the second stage, developers do research and conduct a needs analysis of their users. In order to conduct a needs analysis, designers come up with a number of scenarios. Scenarios are simply situations that users could find themselves in while trying to complete a task using the software. Generally, the more scenarios designers can think of, the more constructive feedback they will get when testing the user interface. There are a number of ways to come up with scenarios. Designers can conduct interviews with users, observe users in the environment in which the software will be used, and explore any existing software that is designed to achieve similar goals. Depending on the purpose of the system, scenarios can be very diverse.

In critical situations, such as those that can be found in hospitals, scenarios need to be taken to the extreme and include situations that are extraordinary. In other, more laid back situations, such as an office setting, scenarios can focus more on normal, everyday tasks.
However, even in an office setting, there should be a few scenarios that are designed to test the limits of the software. For example, if the software system being developed is for a customer service representative, an extreme scenario might be that the customer service representative is having a phone conversation with an angry, yelling client and trying to calm the client, while sifting through papers on her desk to find a document, and looking up some other information on the computer. A main purpose of using scenarios is to make the program as flexible as possible.

Two main aspects of the third phase in developing a user interface are creating specific usability goals for the software and creating an initial prototype of the main component of the system. Prototyping is a key aspect of user interface design. It allows designers to quickly put together a possible layout for screens and test them without having to waste time writing too much code. Prototypes are often very shallow so that when using a prototype it appears that things are happening when you complete task but information is usually not saved. For example, if developing a system for doing inventory, in a prototype it may appear that a user has just added six new items, but the database driving the inventory would not actually reflect the changes.

Stage four of the interface development process is to iterate and refine the interface. More features may be added to the prototype, usability testing is done and based on the feedback changes will be made. Stages two, three and four are probably the most important in terms of creating a usable system.
By the time stages five and six are reached there is little left for the interface design team. They have already submitted their specifications and the coders take over and implement the system. Lastly, in phase six, the software is deployed. Often times the users of the system are provided with training. Phase six also includes any maintenance that will be done on the system such as bug fixing.

3 Designing Children’s software without the help of Children

HCI did not really take off as its own discipline until the early 1980s with the introduction of the Computer Human Interface (CHI) conference. Today, CHI remains one of the most popular conferences in the field. As technology has become more prevalent, developers began targeting software towards children, both for educational and entertainment purposes. The traditional methods of user interface design do not quite translate to developing software for children. Although, often times the age of the target users is considered, children were not considered to be part of the design group.

Early Work in this field was not motivated by designing and creating interfaces that were more usable for children, but instead by what aspects of software children enjoyed using. This was first motivated by people looking into what makes computer games enjoyable for children.
3.1 Evaluating the Appeal of Computer Games

Malone [13] studied an educational computer game of darts, which attempted to teach children about fractions. In the game, a screen is presented to the user with a number line and few balloons scattered across that number line. The goal of the game is to pop the balloons by throwing darts. On each try the user types in a fraction representing the position along the number line at which they would like to throw the dart.

Children were in no way a part of the design process. Design and implementation were completely handled by adults. Instead, the study began after the game had already been developed. This kind of study provides a good beginning for the HCI community in the design of children’s software. As mentioned in the previous section, evaluating existing software is often a helpful step in the design process as it can give designers ideas about both successful and unsuccessful aspects of design.

Using this game of darts as a base, Malone chose eight different features that he thought could be of interest. He split his pool of eighty fifth graders into eight groups. Each group was given a version of the dart program that added one more of the eight features which interested Malone.

Malone measured the appeal of the program by how long students played the dart game. From these results he drew some interesting conclusions. As expected, he was able to decide which aspects of the game students enjoyed most, but he also noticed that the aspects that motivated the girls were different than those that the boys enjoyed. In particular, he was able to conclude that boys liked the fantasy aspect of the game where the arrows popped the
balloons. This feature did not excite the girls much at all. In fact, they disliked the fantasy and tended to play the variation of the game with fantasy for shorter periods of time.

The impact of this study on HCI is not simply noticing that introducing fantasy into the software will probably keep children interested for longer periods of time. Instead, it is the heuristics Malone provides for designing enjoyable user interfaces.

He outlines a few key questions in three different aspects of game design for children. The first set of questions relate to the challenge presented by games. A few sample questions he asks are: “Does the activity have multiple level goals?” and “Does the interface provide performance feedback about how close the user is to achieving the goal?” Pertaining to fantasy he asks: “Does the interface embody emotionally appealing fantasies”. Lastly about curiosity he asks: “Does the interface capitalize on the users’ desire to have ‘well formed’ knowledge structures?”

Malone suggests a number of other questions that are pertinent to enjoyable user interface design as well, but from those presented here, it is evident that Malone began pushing the boundaries interface design to consider children. He did not suggest using children’s input during the design process, but he at least began to recognize that children are a very different user group than adults. He also points out that motivation is key. Software for adults often has a very specific goal in mind – to speed up or complete a task that is usually done at work. In children’s software the typical goal, learning, is much more broad and hard to pinpoint.
3.2 Still neglected

By 1987, five years after Malone’s publication, two researchers from Yale conducted a study on a very similar topic [6]. Instead of computer games, however, their focus was on a program trying to teach math. The intention of the software was for children to learn and practice addition with single digit numbers. Like Malone’s study all investigations of software take place after the programs have already finished development so there is no chance for findings to effect the design of the tested programs.

Unlike Malone’s study, however, the motivation for this study was purely driven to test the quality of the user interface. Frye and Soloway make an important claim that differentiates educational software from other types. They mention that users of educational software are at a disadvantage because they often do not have a background in the domain of the program since the program is trying to teach them about the domain. Often times, background information in the domain of a program is very useful in understanding what goals that program tries to achieve and how it reaches those goals.

The study produced a few critical interface design mistakes that probably would have been caught had children been used to test prototypes of the system. One of the more difficult tasks that an interface designer faces is separating themselves from their own design. Aspects of the program that seem straightforward to the designer are not always straightforward for the user. Given that the designer created the interface, certain features are blatantly obvious to them. It is virtually impossible for a designer to take a step outside of their own design and test it as if they had never seen it before.
As I have fallen victim to this trap a number of times, it seems like the creators of this math program have as well. Frye and Soloway found that one problem with the interface caused some serious problems for the children. For each addition problem, the children were supposed to type in their answer using the keyboard and then hit the return key to "submit" their answers. Unfortunately, especially with the younger students, this caused some confusion. Many did not understand the need to hit the return and would sit and wait for the computer to respond after they typed their answer. In fact, even after being told about the return key 40% of the five-year olds needed to be reminded to hit the return key in a majority of trials.

This type of problem easily could have been avoided by including children in the design process. If the designers had created a prototype of the program and run a few short sessions of user testing (using children as testers), they would have found this problem quickly, and probably could have come up with a solution to the problem with little extra effort. Instead, this user interface problem, along with a few others mentioned in the study affected the instructional value of the program.

4 Children Playing a Small Role in Design

In the early 1990’s more frequent publications regarding educational software began sprouting up in the CHI conference. The studies presented at this conference were really beginning to recognize the issues regarding developing educational software for children.
4.1 Testing

For example, in 1994 Berkovitz [1] presented a study he did while developing a mathematical program geared towards children. In this study researchers actually incorporated children into the design process. A prototype of the system was developed and a few six and seven year old children tested it.

Having children test the system turned out to be successful for the developers. They recognized a few problems and made adjustments to the interface based on the testers’ difficulties with the initial interface. Testing a user interface on potential users of the system is an invaluable source of feedback for designers. It is especially useful for children’s software because adults have a hard time understanding how children think. It gives members of the development team a fresh look at how people will interact with the program. Making sure the testers actually come from the pool of potential users is also important. In the previously mentioned mathematical software for addition and fractions, there probably were users that tested the system but they were not children, so the results would not have been as meaningful.

4.2 More Input from Children

The next logical step in including children in the design process is to consider input before prototypes are developed. Some of the most defining work with this was done by Scaife et. al. at Sussex University [15]. Their publication at CHI 1997 attempted to develop a program for teaching ecological concepts to seven through fourteen year old students. There
had been some work surrounding interviewing children, but Scaife et. al. clearly defined it in their publication.

To include children in the design process before prototyping began, they interviewed a number of students and teachers separately. They asked teachers about how they traditionally taught ecological concepts, and they asked students how they had come to understand of the ecological concepts.

From these interviews designers were able to draw conclusions about what cognitive difficulties they might face as well as possible issues regarding external representation of the material. They also noticed that teachers used a wide variety of approaches to convey information about these concepts and children’s understanding of them was pretty shallow.

Though Scaife et. al. found their approach to be successful, a number of obstacles surfaced. They found that some children did not want to be creative designers. Some were shy and were reluctant to talk to unknown adults. They also found that involving children at earlier stages of design requires a significant amount of resources and effort.

5 Children as Equals

In 1996, the field of HCI embraced the fact that designing software for educational purposes required special attention and created a special interest group at the CHI conference called CHIkids. As stated in [2] the key reasons for creating this group is that “we as designers of future technologies stop and listen, observe and collaborate with children of all ages. It is the only way that we will become sensitive to the needs and desires of one of the fastest
growing populations of technology users in the world: our children.”

A few of the studies mentioned earlier touch on these important aspects of designing interfaces for children, but none of them bring all the pieces together. Druin and her group at the HCI lab at the University of Maryland have taken great strides to improve HCI in educational technology. In a number of publications ([4], [3], [2], [7]), Druin and her team have developed a way to include children in the design process, not only as testers, but also as designers. They call their design process cooperative inquiry.

Druin characterizes four different roles that children can take in the design process: user, tester, informant, and design partner. Each of the roles encourages children to be more involved.

In Malone’s [13] study of computer games we saw children take the role of users. The technology had already been developed and the children were being observed by adults. Two main reasons for children to play the role of the user are to help future developers and to understand the learning process.

We saw children as testers in Berkovitz’s study [1]. The basic responsibility of testers is to test prototypes of software as they are being developed. Testers do not enter the design process until adults have already completed the initial design. By using the children as testers, developers are trying to answer questions such as: “What parts of the technology are confusing?”, “What parts do children like?”, “Where are the bugs?”, and “Can the children learn the technology?”. By getting these questions answered, the developers and designers can make immediate alterations to the software. As testers, children can have a
very strong impact on the design.

Scaife et. al. [15] defined the idea of children as informants in their study. Activities of children as informers include participating in interviews, using existing software, as well as participating in the prototype process. Many “low tech” prototypes are often constructed before moving to the computer. Materials for building these include paper, glue, string, pens, and other general arts and craft materials that most children are very familiar with.

As informants children have a excellent opportunity to significantly effect the outcome of the software being developed. The extent to which they are included, however, is still monitored by the adult members of the design team. Many designers recognize that children have numerous unique ideas that can be invaluable to the designers, but there some that are infeasible to implement in the software.

Druin believes that though each of the previous three roles help developers of educational technology, it is not quite enough. She believes that children should be considered as design partners. By design partners she means that along with being informants children are also equal stakeholders.

Although a few other groups have also developed similar ideas about the role of children in design, what Druin brings to the discussion is her methodology. Cooperative inquiry [3] is the name used for this approach to the inclusion of children. In order to carry out cooperative inquiry successfully, there are three different principles to consider. These principles are all built on top of the foundations of good interface design presented by the HCI community and are mentioned in a previous section.
5.1 Cooperative Inquiry

The first aspect of cooperative inquiry is to create a multidisciplinary team that includes both children and adults. Secondly, field research must be done. When creating an interface for children it virtually impossible to conduct a traditional interview with a child and figure out their software needs. Younger children often have a difficult time expressing this information which is why more extensive field research is necessary. The last crucial part of cooperative inquiry is the use of both low-tech and high-tech prototyping by the multidisciplinary team. This give children the chance to be creative and truly impact the design of the software.

The methodology of cooperative inquiry stems largely from work done in something similar, called contextual inquiry. The main adaptation is that children are actually part of the design team, helping to make suggestions and decisions whereas in contextual inquiry, children are mostly just observed. In cooperative inquiry children are genuine researchers, taking notes, making observations, and true members of the research team. This approach to design seems to parallel the creation of a community of practice [17]. The members of the design team form a community of practice with the goal of designing a piece of educational software. All members work together with the same common goals.

Cooperative inquiry has been put into practice in a number of situations. The first was during the development of a story telling technology called KidPad [5]. The interesting feature of KidPad is the way users navigate through stories that they have created. The design team found that traditional navigation through story telling programs was similar to web browsing. They found that some children would have a difficult time remembering
where in the story they came from or where they were going. They likened moving around to closing their eyes and all of a sudden being in a new place when they opened them again. What KidPad did to reverse this feeling was zoom from one location to another so children could see where how they got from one place in the story to another. This alternative method for zooming, came about because developers followed the cooperative inquiry method.

One of the challenges of cooperative inquiry [4], especially for adults, is that there is no one authority. Children are certainly not in charge, but the adults do not have complete control either. This is a difficult adjustment for both children, who are used to following adults, and adults who are not used to considering children as equals. If this hurdle can be surpassed, cooperative inquiry has the potential to create some wonderful software.

6 Now What?

Since Druin’s publications in 1999, work in the area of HCI and educational software has begun to grow. For example, Hanna [8] took Druin’s work one step further and began outlining some of the finer details of incorporating children into the design process. She addresses issues such as how to motivate the children involved, how to set up the room in which design teams will work, and how to approach different age groups of children.

There are has also been work done in the Netherlands regarding how to compare different usability testing methods [14]. They establish a framework with several indicators of good testing techniques, how to describe the testing technique, and what characteristics of children affect testing methods.
There has also been some work that takes interface design in a different direction. Not only do Wyeth and Purchase [18] take a different design approach, but this group at the University of Queensland also pushes beyond the traditional GUI software towards something more tangible. In this system, called Electronic Blocks, children play with physical blocks, with electronics inside, that combine to create different behaviors. Rather than using some of the design techniques discussed previously, this group found that it was crucial to consult research in the field of developmental psychology.

While Electronic Blocks was not without its interface problems, it appears to be a relatively successful endeavor outside the world of the common GUI. Branching outside of GUIs has the potential to promote a more constructionist software environment. It gives children tools, in this case blocks, to manipulate in order to make discoveries, in this case about electronics.

Also, the group at the University of Maryland has continued to develop new ideas regarding interface development with children. In a publication in 2004, Guha et. al. [7] present a new plan, built on top of cooperative inquiry, called mixing ideas. One of the disadvantages to cooperative inquiry was that some children because disappointed when their specific ideas were not being used. Mixing ideas is an attempt at overcoming this problem.

Similar to cooperative inquiry, children brainstorm and come up with their own ideas. There are then three phases of mixing before reaching the final solution. At the first step, there is an initial attempt to start mixing these ideas in small groups consisting of two or three children. In these groups the children, with the help of adults, were able to combine
aspects of each idea to create one that all participants felt they had contributed to. Then the groups are combined together so that there are two large groups, and each makes a mix of their ideas. Lastly the two groups come together with their two ideas and they are combined resulting in the final product. From post interviews and observations, researchers found that adding the step of mixing ideas was helpful because everyone contributed.

7 Conclusion

It seems that children’s technology and HCI are finally beginning to be recognized together. CHIKids, the part of the CHI conference focusing on children, has continued through at least CHI2003. I tried searching through the CHI2004 and CHI2005 websites but was unable to find any reference to them. Even if CHIKids has ended, I think that the eight years it was in place affected the field of HCI and children’s software. In fact, it seems that the design of educational software has grown into its own conference, which is affiliated with CHI. There is currently a conference called Interaction Design and Children (IDC) [11] [10] [12]. This conference began in 2002 in the Netherlands and is still growing in size.

Two years prior to the first IDC conference, it appears that two out of seventy-two accepted papers at CHI2000 considered children [9]. This neglect of children’s software is no longer the case. At IDC 2003 [9] there were seventeen papers, eleven posters, and five demos. Growing even further, last year at IDC 2004 there were a total of forty papers published about HCI issues as they relate to children’s software.

As evidenced by the increasing amount of publications in the area, it seems that the
two previously separate worlds of HCI and children’s software are finally meshing together. It seems that researchers have finally realized that both adults and children benefit from including children in the design process.

There are certainly some obstacles to this approach, especially when children work with adults as equals. We saw many instances of this hurdle in the constructionist literature.

In the future I think that it will continue to be important include children as members of design teams. This will not only be helpful for traditional GUI style interfaces, but also for generations of interfaces to come. As was touched upon briefly here, there are alternative computer interaction styles, such as the tangible interface exhibited by the Electronic Blocks. Researchers in HCI still have a decent amount of work to do before alternative style interfaces can become mainstream. Since researchers know even less about these alternative styles, if they are going to be applied successfully to systems for children, children will need to play a significant role as a member of the design team.

References


