Journal of Mason Graduate Research
Volume 3 Number 1 Year 2016
© George Mason University, Fairfax, VA
ISSN: 2327-0764
Pages: 22–39

Usability Survey of Educational Software Used by Children at Schools

MOAYAD TAIBAH

George Mason University

The main focus of educational software today is the content it delivers, but little to no care is given to the usability of such software. Interfaces are made colorful and attractive to children only to fail in providing a truly fun and usable experience. The purpose of this paper is to provide the designers with an understanding of the child users of educational software, who are substantially different from adults. In addition, it addresses the school environment where the software is mostly used and provides a better understanding of the educational goals, including several ways the goals can be conveyed and delivered via a computer system. Finally, the paper includes a discussion of the different interface paradigms and what they offer for educational software, specifically, the potential idiomatic interfaces possess for children's mental and physical developmental levels.

Parents care for their children's education and want the best for them. With the advancement of technology and epistemologies (Bichelmeyer & Hsu, 1999), many schools began to introduce computer-based learning to adapt children to computer literacy and introduce them to new, more exciting, and entertaining methods of learning that would increase their retention of information and improve their understanding of concepts. School officials and parents may not accept educational software that do not meet their standards, but what they usually care about is the educational content while not enough emphasis is put on the usability of the application (MacFarlane, Sim, & Horton, 2005). Usability is defined by the International Organization for Standardization (ISO) as: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context" (Abran, Khelifi, Suryn, & Seffah, 2003). The specified users of these applications are not parents nor are they the school faculty and staff, rather they are the children who are considered as the primary users whose needs are to be met. As stated by the definition, to achieve an acceptable level of usability, designers must care for facilitating the achievement of the goals (maximizing the learning outcome with the possibility of entertaining while doing so) intended by the user (young learners with physiological and cognitive capabilities that differ from adults) within the specified context (classroom). With these differences in mind, usability for educational software aimed at children may seem to have the same requirements as software meant for adult use on the abstract level, but they do differ in how they are to be achieved and the considerations taken while developing the user interaction of these applications.

The aim of this paper is to identify the usability factors that differ by the audience, goals, and context of educational software from traditional software applications, to provide general guidelines on usability solutions that should be considered when developing such software. Finally, a new direction for future work is suggested with an interface paradigm that can further increase the usability of educational software and software aimed at children in general.

Catering to the Educational Software Audience

Due to the developmental nature and continuous growth of children's capabilities, especially those who are ages 2 through 11, it is unjust to establish usability doctrines based on the knowledge, experience and capabilities of adults. Educational software should cater to the changing nature of children's physiological and cognitive skills and adapt to their capabilities by establishing easier introductions to the software's features and facilitating its usage. From the age of two to eleven, children traverse through four developmental stages. Starting from the sensorimotor stage, children's stimuli and response are dependent on their senses. They then grow into the preoperational stage, and further to the concrete operational stage, where their cognitive skills begin to become like those of an adult. Eventually, children reach the formal operational stage, where they act similarly to adults (Bruckman, Bandlow, & Forte, 2002). The focus of this paper is on elementary school children aged six to eleven, which aligns with the concrete operational stage (stage 3). This focus is due to the fact that children start becoming more like adults beyond that stage in the psychological and physiological sense. Moreover, the years of education below this stage are too few and may have different focus in terms of goals and needs that need to be met. Finally, children in the concrete operational stage are expected to have relatively fine control over the mouse and keyboard, and basic reading and writing skills that may be required by educational software (Bruckman et al., 2002). It is worth noting that children are rapidly developing during the concrete operational stage, they are getting better at controlling their motor skills and their cognitive functions are in continuous development. In the following sections I address the cognitive and physiological changes of children in the concrete operational stage and discuss how the continuous growth and development should shape the development of the educational software's interaction.

Cognitive Skills

Cognitive capabilities are continuously evolving in children up until the formal operational stage. However, in the concrete operational stage, a substantial gap between children's and adults' cognition remains, affecting the information processing speed (Hourcade, 2003) and the ability to understand abstract concepts and form hypotheses (Bruckman et al., 2002). These developing capabilities affect several control capacities in children, especially those that involve motor skills, as will be explored later in the article, and affects their use of words, whether it be in reading or writing (Schneider, 1996). Moreover, many children struggle to overcome barriers of entry in software due to less tolerance and shorter attention spans than adults (Bruckman et al., 2002; Chiasson & Gutwin, 2005). In order to bridge such a gap, the user interface built for children should reduce the use of complex movements, textual commands and text rich interface, and increase the abundance of visual cues and concrete examples that explain abstract concepts.

Many of the adult interfaces contain a large amount of text, whether they are presented as part of the interaction (such as command lines), informational cues, or as help. Children in the concrete operational stage have learned to read and perhaps write, but their skills are not as developed as adults' skills, which may lead to misunderstandings and incorrect interpretations from both the children and the software they are interacting with. Chiasson and Gutwin (2005), for example, suggested keeping textual elements at minimum, and when required, they should be accompanied with alternatives such as having the text script read to the child or providing a graphical representation that conveys the same meaning.

Abstract concepts and functions may be hard for children to grasp, which may pose yet another cognitive barrier that should be overcome, but they have a better understanding of the concrete. Children in the concrete operational stage may get confused if a general concept is explained to them, but once concrete details are given they gain a better understanding of the function that lies underneath. An example of this is a library interface that went from the abstract concept of active queries, which changed the results with each change, to the concrete graphical interface that provided a metaphorical way for children to search for animals (Druin et al., 2001). Encasing the abstract concept with a context-specific, concrete function may increase the number of interface elements required for it to be firmly comprehensible and against the guidelines that suggest keeping the interface as simple as possible. However, it is a way that shows promise as a solution that aids in overcoming the barrier of confusion children have towards abstract concepts.

Children also have a shorter span of attention, which means they may either lose interest if they do not find value in what they are doing, if they don't know what to do, or have forgotten the task at hand (Chiasson & Gutwin, 2005). Interfaces need to provide immediate feedback, remind them of the task at hand if they seem to be distracted (perhaps by monitoring the time since last action), and provide them with easy on-boarding into the application.

Physiological and Motor Differences

One of the factors that affects the children's interaction with regular software interfaces is their ongoing physical development (Schneider, 1996). Physical capabilities of children affect their usage of input devices and interaction with the interface. The following segments discuss some of these impediments and ways to overcome them in order to provide a better interaction experience for children.

As pointed out earlier, children have slower information processing speed than adults and, thus, require more time when clicking on a smaller target, according to Fitts' law (Hourcade, 2003). This may lead children to experience frustration that is not related to the content of the software, but by the hindrance caused by usability issues that come between them and achieving their goals. Fitts' law takes the size of a clickable object and the distance between the pointer and the target into consideration. Therefore, increasing the target's size and grouping similar objects that have a tendency of being used together can reduce movement time for clicking the target. However, it may reduce the screen space as a consequence of the increased icon size, which can be a precious asset, especially for mobile and hand held devices, which, in turn, would reduce the amount of actions a user can take on the screen. Although children differ in the level of control they can handle, reducing the complexity of an interface and the number of actions available to a child may prove advantageous, as it would provide them with a better direction to achieve their goals (Hourcade, 2003).

Another issue that is affected by the children's motor skills is the use of drag and drop actions when using an interface. Based on recent research, it is easier for children (ages 9 through 13) to use a point-and-click action instead of dragging and dropping an interface element (Inkpen, 2001). In the study, children were able to perform better and were more satisfied with using a point-and-click mouse gesture for an educational game in which they had to move items from one part of a screen to another (Inkpen, 2001). This is due to the fact that many children have trouble holding the mouse button for long periods of time (Bruckman et al., 2002; Strommen, 1994). Previous research also suggests that the selection of multiple objects, which can be accomplished by the drag and drop action, is hard to achieve for children (Inkpen, 1997).

Adapting to the Constant Development in Children

Children's capabilities are continuously growing in the concrete operational stage. For example, a child at the age of 9 may lack any interest toward a software application catering to a child at the age of 7. As a result, children may outgrow the interface that was initially built for their capabilities and even lose enthusiasm for using the software (Hourcade, 2008). However, developing an interface that suits the needs of children within the whole concrete operational age range is not a viable solution. Even though some of the cognitive and motor capabilities begin to develop at this stage, these functions continue to escalate dramatically in children from year to year if not faster. One of the proposed solutions to this dilemma is to design for the minimal intended age of use, providing aid that fits their motor and cognitive level (Hourcade, 2003). Another researcher suggested providing a customizable interface that provides minimal interaction items that suits the youngest of age and the older children can then modify the interface and increase its complexity to better suit their capabilities. Each solution has limitations. Providing a minimum barrier of entry interface may be easy to implement, but older children may grow bored quickly and are more likely to stop using the application. The other solution of providing a customizable interface may require additional implementation effort and can prove too complicated for older children to modify, as they need to know what can be modified before being able to modify it. A more suitable solution may be to provide an evolving interface, in which complexity and functionalities are unlocked depending on the children's performance (Hourcade, 2003).

Aesthetics and Interface Design Guidelines

A lot of the day-to-day interfaces and devices that children use while interacting with an educational software are built for adults, whether they are hardware devices or user interface elements, such as icons and font. Although children are able to adapt to these specifications, they may spend a considerable amount of time and effort to overcome this threshold. As a result, several researchers conducted experiments with different aesthetics and interface aspects that may provide better on-boarding to children's interaction with software (e.g., Jones, 1993; Schneider, 1996; Uden & Dix, 2000). Although these experiments built on broader software design concepts concluded from research with adults, they focused on children and their developing capabilities.

Fonts and Textual Representations

Children's reading and writing skills are affected by their developing cognitive functions, which affects the understanding of orthography (Schneider, 1996). As a result, text-dependent interfaces, whether they are command-based interfaces or text-rich displays, need to be kept at a minimum. However, it is hard to provide a software application, let alone an educational application, that is detached from text, either as an indicator of actions or as an informative medium. As such, Bernard, Mills, Frank, and McKown (2001) conducted an experiment to determine the font size and type children preferred in the software they used. The experiment was conducted with children in the age range of 9 to 11 years. The authors concluded that children prefer, in terms of readability and attractiveness, fonts of size 14 and of Comic font type, with Arial as the second most preferred font type. In addition to font type and size, the length of textual lines should also be taken into consideration. In a similar study focused on the number of characters per line (CPL), Bernard, Fernandez, and Hull (2003) did not find a significant performance difference when children, aged 9 to 12, read passages with different CPL, but children reportedly preferred passages that were short in length (around 45 CPL).

Icons

Icons and metaphors can be used to replace textual menus and tabbed actions in a children's interface to reduce the interface's complexity, but several factors should be considered when choosing an icon. Icons depend on the users' mental model, which is the cognitive shorthand explanation of a concept, in order to gain meaning and hint at a certain function (Cooper, Reimann, & Cronin, 2007). Icons may become detached from the designer's original intentions of how it should be interpreted and thus it is advised to keep the end users in mind when designing them. Mental models are also built upon previous experiences (Inkpen, 1997). For example, the computer folder icon is based on an office folder. Children in the concrete operational stage have likely not had as many life experiences as adults, and, so, may not understand adult metaphors (Bruckman et al., 2002; Inkpen, 1997).

Other researchers, such as Uden and Dix (2000) and Jones (1993), tried to identify additional guidelines for creating icons suitable for children as the main audience of the software. Icons are not just meaningful images, they are used to convey the functionality and intention of what the image represents, and so images must be chosen carefully so that their meanings are not confused with other intentions. In the small space given to an icon, the designer must deliver three aspects: function, concept, and representation (Uden & Dix, 2000). Function is the underlying mechanism that will be run when the icon is interacted with directly, and causes an immediate reaction. Concept is the overall context in which the icon exists in, the higher-level goal that this icon and other elements around it convey. Representation is the external image of the icon, and it must embody both the function and concept. To satisfy these criteria, the shape, size, and color of the icon must be chosen carefully. For example, in an arithmetic calculator application, the most frequent actions (the arithmetic operations) would be bigger in size, similar in color and have symbols that match the operations they would execute. On the other hand, the icon used to exit the program is used less frequently and has a major consequence on the application (terminating it), which is why it is smaller in size, colored in red, and has an icon that represents the termination of the application.

Several studies indicate that animated icons provide a better explanation of the actions they denote (Jones, 1993; Uden & Dix, 2000). Static icons provide an abstract state of a possible action, which can be misinterpreted, whereas an animated icon shows the complete sequence of the action associated with the icon. Uden and Dix (2000) concluded that children, aged 5 to 7, are much more likely to recognize the function of an animated icon over a static icon, which concurs with findings in Jones' (1993) similar experiment with elementary-aged students from three age groups (6, 8, and 10). The children also found the animated icons more enjoyable and the icons captured their attention and imagination (Uden & Dix, 2000).

The School Environment and the Context of Use

Schools lessons are often given to groups of children who are of similar age where the teacher lectures, and students listen and participate from time to time. With the introduction of electronics, the level of collaborative interactions can be heightened and the role of the teacher can change from a lecturer to one who gives guidance and assistance when needed. This is in contrast with the typical applications that adults use at home or in the workplace, where they are usually working solo on the software (Stewart, Raybourn, Bederson, & Druin, 1998). Providing educational software that allows for collaboration and assistance when needed may prove useful to solving and overcoming time and interaction constraints and sustaining a more usable interface that is suitable for the educational environment. Collaboration between children can take one of two forms, either using two separate computers and communicating remotely via chat, voice, video, or a combination of the previous choices, or it can be done by sharing a single device and communicating verbally. The second method of communication more closely resembles the traditional school environment and may be more useful in an educational setting (Inkpen, Booth, Gribble, & Klawe, 1995). Research suggests that sharing a single device by two or more children can be more beneficial for them than working on separate devices (Stewart et al., 1998). However, other researchers found children significantly less productive when sharing a single input device than when each child had an input device of their own (Inkpen et al., 1995). Having a single shared device (usually referred to as Single Display Groupware) resulted in a fruitful collaboration session, as children, aged 8 to 12, provided and asked for help more willingly. Using a single display groupware also resulted in better exploration of the interface because it peaked the children's curiosity and they swapped input devices to interact with different elements of the interface. Inkpen et al. (1995) found that children shared the computer device differently based on their gender. The researchers implemented a sharing system in which the child who had control could give control to the passive player or the passive player could take the control from the active player. The results showed girls solved more puzzles (the goal of the game they were given) using the “give protocol” than they did with a single mouse. The boys, on the other hand, solved more puzzles using the “take protocol” and significantly less using the “give protocol,” when compared to the single mouse control.

Goals of Educational Software

Gaining an understanding of the goals of an educational system aimed at children provides a perspective on what needs to be taken into consideration when resolving usability and design issues. It also allows understanding the goals from the viewpoints of the users, whether they are students or teachers. Educational software is one of the adaptations that build on the traditional educational method in which concepts are delivered via a computer system, and thus it shares the same goals of said educational methods, which can be broadly summarized as providing the maximum learning outcome and self-actualization that the students can attain. In addition to these goals, educational software applications, especially those aimed at children of the concrete operational stage, also include the goal of providing a fun experience.

Maximizing the Learning Outcomes

One of the goals of educational software is to maximize the learning outcome for students, which can be achieved by increasing the knowledge circle, incorporating new concepts, learning their processes, and providing training and exercises to retain realized concepts. However, this is in contrast to the goals of traditional software, where the outcome matters the most, regardless of the process used to achieve it. As a result, traditional software aims at maximizing the speed of performance. In contrast, the goal of educational software is to teach new concepts or confirm an existing one to the user, in this case a child. There are several aspects to consider when providing a learning platform, such as catering to the goals of both teachers and students, and the methodology by which the concept is presented (Bruckman et al., 2002).

Teachers need to be able to monitor and measure their students' progress toward mastering taught concepts. Educational software can cater to the teachers' needs by providing a usable interface that measures the performance of their students (Bichelmeyer & Hsu, 1999). The software may also allow teachers to interact with students when they need help in advancing through any of the learning concepts and perhaps even allow for one-on-one interaction via the Wizard of Oz technique (Steinfeld, Jenkins, & Scassellati, 2009), where the teacher impersonates a computer when interacting with the student. The interaction can be live, where the teacher adjusts the content depending on the current performance of the student, or it can be prepared for the students based on their previous performance.

According to Bruckman et al. (2002), there are five means by which an educational software can be utilized for providing educational content, where each of them dictate a different category of behavior from the software. The first of the categories is computer acting as a tutor, in which the student is provided with the information and quizzed on it later on. This category tracks the student's progress and evolves accordingly. The second category is computer as a tool. In this category the software acts as a tool for students to use and experiment with different inputs, which encourages the construction and the exploration of concepts. An example of the aforementioned category is a weather forecasting tool that allows children to input different aspects that affect the weather and watch the outcome. The third category, computer as a tutee, encourages constructed understanding of concepts as well by having the student program the computer and learn from the resulted creations. Distance education, using online environments to emulate a classroom, and information sharing all fall within the fourth category: computer supported collaborative learning. This category stimulates a collaborative environment of social constructivism in which students present their ideas and learn from each other, providing an opportunity to brainstorm and build new ideas from old ones. The last of the categories is educational learning, sometimes referred to as edutainment (Bruckman et al., 2002). Software in this category mixes the entertainment experienced from computer games with learning objectives by weaving these objectives with a game-like narrative that can be more informal when compared to the software based on traditional pedagogies (Okan, 2003). The last category will be addressed and assessed in detail in the next section.

Fun in Educational Learning

Children often have short attention spans and, if something does not appeal to their tastes, they may quickly throw it aside and shift their interest towards something more engaging (Chiasson & Gutwin, 2005). Edutainment allegedly provides the means to combat the children's quick judgment by providing a captivating and interactive experience (Okan, 2003). The goal of such software is to provide entertaining methods of learning information, promote self-development, and motivate children to understand and learn about different concepts. However, Okan (2003) also argued that educational entertainment may be destructive to the learning process and does not present true educational information to children because it emphasizes the entertainment factor more than the educational, which may reduce the weight of educational value in the eyes of the children using the software. In addition, it sends a message that if an educational concept is not fun, then there is no learning activity occurring. Lastly, Okan argues that edutainment provide a shallow layer of information and discourages further exploration and metacognition. Indeed, MacFarlane et al. (2005) did not find a correlation between children having fun and learning. However, they did find a correlation between fun and usability. Similarly, Tractinsky, Katz, and Ikar (2000) found a positive correlation between the aesthetics of an application and its usability. In the experiment, participants who thought an interface was superior in terms of aesthetics also considered the interface superior in usability before and after using it. Currently there is limited research comparing fun and usability, but it is possible that the user's positive thoughts about an application's level of fun may lead to better performance and a more forgiving judgment of the application's actual usability.

Future Work

Although the studies presented in this article point to the positive results of providing a better user experience for children using educational software, none of the researchers theorized about the possible advantages of employing idiomatic interfaces, including the ability to provide a better and more suitable representation of interface elements and interactions. Idiomatic interfaces have several advantages over the implementation-centric and metaphoric design paradigms commonly employed, specifically for users with fewer real-life experiences with objects and other people, such as the children in the concrete operational stage of development.

Implementation-Centric Interfaces

Implementation-centric interfaces are based on implementation code of the software product, in other words, how the software works (Cooper et al., 2007). Interfaces such as these are not suitable for most of the computer-using population, unless the users share the technical expertise of a designer or developer (Cooper et al., 2007), and even more so for children due to the paradigm's complete disconnect from the users' mental models and high level of abstraction (Bruckman et al., 2002). Children in the concrete operational stage often lack the knowledge and understanding of the details of the product's implementation. As a result, they often cannot relate to nor form an understanding of an implementation-centric design that presents the functionality of the software. Turning a specific feature on and off, for example, is solely based on implementation details, such as using the 0 to represent off and 1 to represent on.

Metaphoric Interfaces

Much of the literature in the fields of Human Computer Interaction and Child Computer Interaction includes discussions about the provision of a metaphoric (including iconic) interface with suitable modifications that adapts to the children's cognitive and physiological capabilities (Cooper et al., 2007; Inkpen, 1997; Jones, 1993; Schneider, 1996; Uden & Dix, 2000). However, most of the research does not cover some of the more critical aspects and difficulties faced in achieving relatable metaphoric elements. Metaphoric interfaces possess the merit of being intuitive in the sense that they build on the users' previous experiences and interaction with similar elements thus easing the process of absorbing new information. However, this enforces the powerful dependency on having preexisting knowledge of a similar concept or element, whether from real-life or computer-derived experiences. This dependency may prove disadvantageous for children as they have not experienced as much as adults and may lack the mental model that relates to the designer's intentions for the metaphor.

Inkpen (1997) describes the metaphor of a file folder, a familiar computer icon, that adults found clear and relatable, but children did not find as clear. File folders are frequently used by adults in offices or at different work locales, which enables adults to form a hypothesis on the file folder icon’s functionality (storing files) in computer systems. For children, however, file folders may be foreign and unfamiliar objects that they have yet to experience. As a result, children have to either intuit the icon’s function or learn the function via a brief tutorial. A solution to this challenge is to limit the metaphors to the domain of knowledge that the children possess. For the previous example, a school bag can be used as a placeholder of objects instead of a file folder. However, this may exponentially increase the difficulty of designing an interface, as metaphors are not easy to come by (Cooper et al., 2007), even more so for those metaphors suitable to the budding experiences of children.

Metaphors are also bound by time. For example, a common metaphor for saving a file is the icon of a floppy disk, which has not been widely used for some time. Future generations may struggle to form a relation between the disk icon and the saving function because they have not experienced the use of nor seen an actual floppy disk. Overcoming this limitation may not be feasible as the latest technological advances move toward more simplistic exterior representations (e.g., televisions and tablets in the Appendix), resulting in icons that can be confused for multiple objects. Old objects, however, retain their uniqueness in shape and figure and can be identified more easily (Uden & Dix, 2000).

The difference in cognitive development between children and adults can also make it more difficult to design appropriate metaphors for children. Designed metaphors depend on the concurrence of the designer's and children's mental models (Cooper et al., 2007). To enact such concurrence, the designer has to be properly acquainted, through experience, with children's cognitive reasoning or include children as participatory designers during the design process (Bruckman et al., 2002). Both of these solutions require that the designer be comfortable around children and able to extract and correctly interpret their intentions.

Metaphors are susceptible to causing cognitive dissonance. For instance, cultural context may bias the designer's representation of a metaphor toward what is known within that designer’s community. Representing a functionality with a local brand may be understandable by users who are in the same environment and are exposed to this cultural reference. However, it could be the cause of cognitive dissonance to anyone outside of that environment. Children may also experience cognitive dissonance if the metaphor deviates completely from the way its physical counterpart is used. Children may continue using the metaphor as they would the physical counterpart and their confusion may grow due to the lack of expected response, which may lead them to abandon their efforts or await direct instruction to learn the correct usage. Such cognitive dissonance can be avoided by either changing the metaphor completely, or adapting the complete element to the real-life object and, in doing so, limiting its capabilities to that of the real-life object, which can be counterproductive for some cases (Cooper et al., 2007).

Metaphors may resolve the need of gradual understanding behind the meaning of an interface element or interaction, drastically reducing the time required to learn navigating an interface, but it becomes a hindrance in the way of the intermittent user's usage of the software. Metaphors are ultimately derived from physical real-life objects to represent their technological counterparts and make them relatable to the user. Although this may provide an initial understanding of the purpose of the interface elements, it may cause some inconvenience as the user becomes versed in using the element. An example of such a case is if a calendar is adapted completely as the real-life counterpart where a user can only view a single day or month at a time instead of simultaneously viewing all days in a month or a couple of consecutive months.

There are times when metaphoric interfaces present themselves as an ideal solution and, in such cases, designers should not be reluctant in utilizing them, especially if they prove successful with the intended audience. The animal query interface, created by Druin et al. (2001), is an illustration of such success. The interface is completely metaphoric, providing search filters in the form of locations and habitats, visual keywords, and two cartoon characters that aid in finding the animal in question. The display first provides habitats of animals within the database repository. Children choose the habitat chip of the animal for which they are looking and more chips appear to further define the animal's characteristics (e.g. "what they eat" chip). With every selection, the characters (represented as a boy and a girl) hold the query chips to show the path that leads to the displayed results. Children successfully generated queries using the prototype of the software application and were able to construct more complex queries compared to the traditional method.

Idiomatic Interfaces

According to Cooper et al. (2007), idioms consist of three building blocks: primitives, compounds, and idioms. Primitives are basic functions such as clicking, pointing, and key presses. Compounds are a combination of primitive actions, such as double clicking. Idioms are complex, context-specific functions that are built from a compound functions. Idioms are understood due to their simplicity and uniqueness, rather than based on connections with experiences or common sense (Cooper et al., 2007). Therefore, idioms may provide the means to adapt computer interfaces to children's capabilities with adequate and minimal training provided. In addition, idiomatic interfaces may support educational goals in an educational context, while addressing the shortcomings of metaphorical interfaces.

Based on the idioms' building blocks, it can be inferred that idiomatic interfaces gain their power by limiting available actions to a small number that can be combined to perform complex interactions. This allows for an interface that is appropriate for the cognitive abilities of a child in the concrete operational stage. Additionally, compound actions can be separated into different steps such that each step is a single primitive action. As an example, instead of dragging and dropping to move an item, which consists of two primitive actions (holding the mouse button and dragging) children could click the item once and then click on the location in which they wish the item to be transferred (Inkpen, 2001). This modification achieves the intended goal, moving an object, without compromising the interface usability.

Adapting idioms to children’s capabilities is but the first step to achieving a usable educational software. Although they are not hard to master, idioms still require initial training to learn them for the first time. Romeo, Edwards, McNamara, Walker, and Ziguras (2003) found that children easily understood and adapted to the use of the computer mouse, as opposed to the keyboard, and adapted to the touch screens just as well if not faster. The authors also found children's performance using a touch screen comparable to adults when they observed and analyzed their usage of touchscreens. Both the mouse and touch screens are examples of idiomatic interfaces that require introductory learning, but can be mastered at a rapid pace. However, children may still require easy orientation when they are introduced to a software, especially one with an educational intent, as they may have short attention spans and may shift their focus toward something else if they do not immediately understand the software. One way to overcome this threshold of idioms is to provide affordance, or visual cues, to indicate the action(s) that the user should or could use with the interface elements (Norman, 2002). Idioms can be designed such that their affordance indicates the way in which the child should interact with them. Using the previous idiomatic example of the drag-and-drop function, the interface item could be three dimensional to indicate that it is clickable. Once clicked, a box range can be highlighted and brought forward to indicate the area to which that element can be moved.

In the educational context, idioms can be optimized to support communication between students as well as between students and their teachers. Idiomatic elements are not dependent on a preexisting mental model in the children's minds, they do not depend on a culture, nor are they tied to a specific time. These factors make idioms effective tools that facilitate communication because all the users likely have the same level of knowledge required to construct the different primitive blocks required to use the elements effectively. Idioms reduce the time required to learn how to navigate and use software. This makes idioms the ideal paradigm for school contexts because the limited time available in class can be focused more on learning new concepts instead of the syntax and workings of using the software itself, allowing for the achievement of the maximum learning outcome goal.

Conclusion

It is important that educational software designers remember the capabilities of children in the concrete operational stage. The metaphorical interface is the leading paradigm that currently overshadows the children's software market. Although there are instances when this paradigm is ideal, there are software applications relying on metaphorical design due to the lack of a better solution. However, the idiomatic paradigm shows potential of being a more suitable paradigm to address children's cognitive and physical capabilities. Metaphoric interfaces may be suitable, but extra attention should be spent while developing them to assure that they reflect the children's mental model, not just the designers' model. This can be achieved via participatory design where the children are co-designers in the process. As for the idiomatic interface, there are not many examples of educational interfaces designed with the paradigm in mind and so it is yet to be proven if it is appropriate for children in the concrete operational stage. However, idiomatic interfaces do show promise in solving problems faced by the continuous process of development in children and overcoming the limitations of metaphoric design. This does not mean that the idiomatic paradigm is superior over the metaphoric paradigm. Should a metaphoric element be appropriate in the educational context and well understood by children (e.g. the aforementioned animal search repository), it should be utilized and favored over idioms because no additional training is required to use the metaphor.

Appendix

Similarities between a modern television and a tablet icon

References

Abran, A., Khelifi, A., Suryn, W., & Seffah, A. (2003). Usability meanings and interpretations in ISO standards. Software Quality Journal, 11, 325-338. doi:10.1023/A:1025869312943

Bernard, M., Fernandez, M., Hull S., & Chaparro B. S. (2003). The effects of line length on children and adults' online reading performance. Proceedings of the Human Factors and Ergonomivs Society Annual Meeting, 47, 1375-1379. doi:10.1177/154193120304701112

Bernard, M., Mills, M., Frank, T., & McKown, J. (2001). Which fonts do children prefer to read online. Usability News, 3. Retrieved from http://usabilitynews.org/which-fonts-do-children-prefer-to-read-online/

Bichelmeyer, B. A., & Hsu, Y. C. (1999). Individually-guided education and problem-based learning: A comparison of pedagogical approaches from different epistemological views. National Convention of the Association for Educational Communications and Technology, Houston, TX. Retreived from http://files.eric.ed.gov/fulltext/ED436135.pdf

Bruckman, A., Bandlow, A., & Forte, A. (2002). HCI for kids. In A. Sears & J. A. Jacko (Eds.). Handbook of human-computer interaction (pp. 33-46). Boca Raton, FL: CRC Press.

Chiasson, S., & Gutwin, C. (2005). Design principles for children's technology (Technical Report HCI-TR-05-02). Retrieved from Computer Science Department, University of Saskatchewan website: http://hci.usask.ca/publications/2005 /HCI_TR_2005_02_Design.pdf

Cooper, A., Reimann, R., & Cronin, D. (2007). Metaphors, idioms, and affordances. In C. Webb (Ed.). About face 3: The essentials of interaction design (pp.269-284). Hoboken, NJ: John Wiley & Sons.

Druin, A., Bederson, B. B., Hourcade, J. P., Sherman, L., Revelle, G., Platner, M., & Weng S. (2001). Designing a digital library for young children. In Proceedings of the 1st ACM/IEEE-CS joint conference on digital libraries (pp. 398-405). New York, NY:ACM. doi:10.1145/379437.379735

Hourcade, J. P. (2008). Interaction design and children. Foundations and Trends in Human-Computer Interaction , 1, 277-392. doi:10.1561/1100000006

Hourcade, J. P. (2003). It's too small! Implications of children's developing motor skills on graphical user interfaces (Technical Report UMIACS-TR-2002-104, HCIL-TR-2002-24). Retrieved from Computer Science and Engineering , University of Maryland (College Park, Md.) website: http://drum.lib.umd.edu/bitstream/handle/1903/1245/CS-TR-4425.pdf

Inkpen, K. M. (2001). Drag-and-drop versus point-and-click mouse interaction styles for children. ACM Transactions on Computer-Human Interaction (TOCHI), 8, 1-33. doi:10.1145/371127.371146

Inkpen, K. (1997). Three important research agendas for educational multimedia: Learning, children, and gender. In AACE World Conference on Educational Multimedia and Hypermedia Vol. 97 (pp. 521-526). Calgary, Canada. doi:10.1.1.89.6739

Inkpen, K., Booth, K. S., Gribble, S. D., & Klawe, M. (1995). Give and take: Children collaborating on one computer. In CHI '95 conference companion on human factors in computing systems (pp. 258-259). New York, NY: ACM. doi:10.1145/223355.223663

Jones, T. (1993). Recognition of animated icons by elementaryaged children. Research in Learning Technology, 1. doi:10.1080/0968776930010105

MacFarlane, S., Sim, G., & Horton, M. (2005). Assessing usability and fun in educational software. Proceedings of the 2005 conference on interaction design and children (pp. 103-109). New York, NY: ACM. doi:10.1145/1109540.1109554

Norman, D. A. (2002). The psychopathology of everyday things. In. The design of everyday things (pp. 9-11). New York, NY: Basic Books.

Okan, Z. (2003). Edutainment: Is learning at risk?. British Journal of Educational Technology, 34, 255-264. doi:10.1111/1467-8535.00325

Romeo, G., Edwards, S., McNamara, S., Walker, I., & Ziguras, C. (2003). Touching the screen: Issues related to the use of touchscreen technology in early childhood education. British Journal of Educational Technology, 34, 329-339. doi:10.1111/1467-8535.00330

Schneider, K. G. (1996). Children and information visualization technologies. Interactions, 3, 68-73. doi:10.1145/234757.234765

Steinfeld, A., Jenkins, O. C., Scassellati, B. (2009). The Oz of Wizard: Simulating the human for interaction research. In Proceedings of the 4th ACM/IEEE international conference on Human robot interaction (HRI '09) (pp.101-108). New York, NY: ACM. doi:10.1145/1514095.1514115

Stewart, J., Raybourn, E. M., Bederson, B., & Druin, A. (1998). When two hands are better than one: Enhancing collaboration using single display groupware. CHI 98 conference summary on human factors in computing systems (pp. 287-288). New York, NY: ACM. doi:10.1145/286498.286766

Strommen, E. (1994). Children's use of mouse-based interfaces to control virtual travel. Proceedings of the SIGCHI conference on human factors in computing systems (pp. 405-410). New York, NY: ACM. doi:10.1145/191666.191803

Tractinsky, N., Katz, A. S., & Ikar, D. (2000). What is beautiful is usable. Interacting with Computers, 13, 127-145. doi:10.1016/S0953-5438(00)00031-X

Uden, L., & Dix, A. (2000). Iconic interfaces for kids on the Internet. In IFIP world computer congress (pp. 279-286). Beijing, China. Retreived from http://www.hcibook.com/alan /papers/kids-icons-2000/