Symbo e-mail for small children

Stian Andorsen, Gunnar Hartvigsen and Kai Even Nilssen
Department of Computer Science, University of Tromsø
N-9037 Tromsø, Norway
stian@pronto.tv gunnar@cs.uit.no kain@cs.uit.no

Abstract
In the Symbo project, our goal has been to offer electronic communication to children that cannot read or write. To enable communication between these children, we have developed our own language called SymboL. It consists of around one hundred visual symbols that the user can freely combine when constructing their message. We found that to take the special needs of the children into account one must apply a methodological approach that focuses on usability. It is important to involve the children from the start of the process, and to base design decisions on user interface findings for children. Based on these findings we constructed the Symbo email client. This client has been evaluated in terms of ease of learning, satisfaction and communication.

Keywords: Software for children, symbolic language, symbolic communication, symbolic email, visual language

1. Introduction
Most people find electronic communication through e-mail valuable for both official and private use. A text-based e-mail system requires of course that the user can read and write. For children, age 4 to 8, this is very often not the case. In this group we also find adults with read/write disabilities. In the Symbo project, our goal has been to open up the possibilities of electronic communication to such a user group. Based on an assumption that children, age 4 to 8, are able to communicate electronically using a simple visual language, our project goals have been to develop a simple symbolic language, SymboL, and to construct a dedicated e-mail system.

The project has been conducted in three steps. The first step was to develop a visual language (SymboL). This has been done in close cooperation with children. The second has been to propose design principles for the user interface for children. Children as software users are something of a special case. They are rarely task oriented, or have jobs to perform; they explore and play with technology [6]. They also have difficulties expressing themselves verbally [5]. This renders most system analysis techniques and design methodologies unsuited at best. Developing a user interface that is tailored to children was therefore an important goal. The third step was to build an email system (Symbo) that uses the symbolic language (SymboL) and the user interface design principles.

This paper presents the theoretical framework, method and approach, system requirements, design, implementation and testing of the symbolic language SymboL and the Symbo email client.

2. Software for children
Developing software for children should in theory not be different from doing so for anyone else. However, this is not the case. Unlike grown-ups, it is very difficult to get

inside the children's world. They have problems expressing their thoughts and opinions verbally [5]. Children also have a different mental model and perspective on things. As mentioned, they are rarely task oriented, or have jobs to perform; they explore and play with technology. This means that you cannot simply treat children as "small adults", as is often the case.

There are also many myths and assumptions of what children like and need in computer software. Developers tend to ask teachers or parents, instead of talking directly to the children. This means that children seldom get the chance to voice their opinion, and when they do, it is usually not taken seriously. After all, we have all been children once and have memories about what we did and did not like. Many developers also have children of their own, and feel they know what they need to know based on their experience with their own children. However, such personal impressions may not be valid or, even useful, when developing software for children of today [5].

The goal is to satisfy the whishes and needs of children by giving them the opportunity to contribute in the development process, and make it an engaging experience [2]. Achieving this goal can be difficult. Involving children in the development process is demanding, and usually several months are spent in the pre-production phase before any technology is made [7].

3. Related work

The most widely used symbolic language today is Blissymbolics. It is a communication system originally developed by Charles Bliss for the purpose of international communication. The original target group was children with physical disabilities. The language is composed of over 2000 graphic symbols, has an extensive grammar, which allows for sentences in past, future and present tenses [1]. The price of the flexibility it offers is complexity. Unlike SymboL, the language cannot be used based on intuition, and requires training. There exist several Internet communication clients that use Blissymbolics, but we where unable to find any that was directed at young children.

The ALDICT project has developed an e-mail client (Inter_Comm) based on symbolic communication sets. Users with intellectual or learning disability can compose and send symbolic messages in a graphic environment. The recipient receives the message in his symbol set and language. The system is far to complex for our user group.

The Elephants Memory is a visual language consisting of more than 150 combinable symbols (referred to as pictograms and ideograms). It is oriented primarily towards children, with the goal of providing an exploration tool that enables new approaches to the concept of language [10].

Symbols are used at least in two very popular everyday appliances, though not to the same extent as in Symbo. For instance, many cellular phones enable you to send messages that contain a picture. Another device, which many children no doubt will be familiar with, is the Nintendo GameBoy, which comes with e.g. Pokemon games. It allows children to exchange Pokemon symbols between to GameBoys by using a cable.

Microsoft Outlook is one of the most popular email clients available today. It offers a rich set of features and is very flexible. However, the complexity makes it difficult or even impossible for young children to use. A simple task such as addressing an email requires several clicks and complicated dialogs to complete. In addition, it does not have built-in support for any other communication form than text.

Objective Voice Email for Kids is a plug-in for Microsoft Outlook. It allows you to record and send voice messages, and hence makes email more available for children who

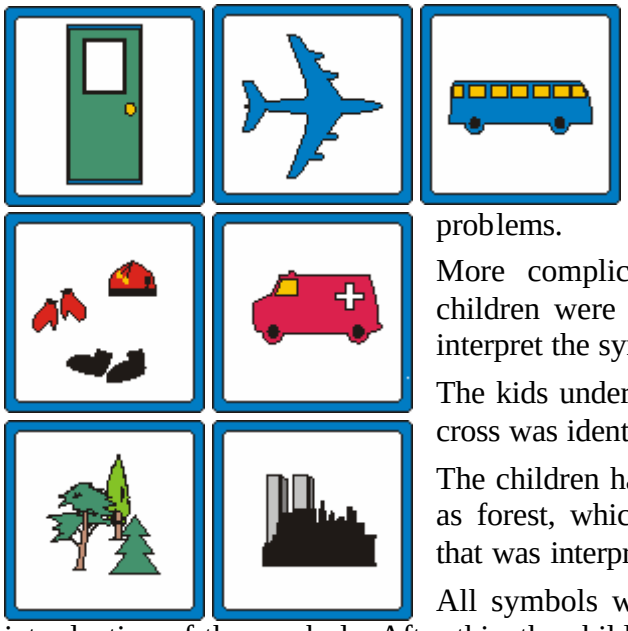
cannot read or write. Nevertheless, it has all the other drawbacks that Outlook suffers. For instance, the interface is complicated and still relay on text to communicate much of the functionality. Another point is that the attached sound files may not be readable by all email clients, and the parents may not be comfortable with the children having access to the default Outlook email contact list.

Unlike Objective Voice Email for Kids, the Pronto Family Email package is a stand-alone product. The interface is adapted to children, but as the other systems it also relies on the ability to read or write in other to use the interface. The main communication form is text, but sound and video is also available.

The market for children's software has exploded the last few years, and there are countless packages available. However, these are not covered in this paper. This is mainly because the usage pattern in our client is quite unique as it is neither a game nor an educational tool.

4. The symbolic language Symbol

The symbolic language SymboL has been developed in close cooperation with children in the potential user group. In spring 2001, comprehensive tests were conducted together with children at Workinnmarka primary school (5 kids from the junior class, 6-7 years old) and Gimle kindergarten (preschool) (five of the eldest kids (5 years old)) in Tromsø, Norway.



The children had few problems recognizing most of the symbols that were presented to them. The simplest symbols, such as door, airplane, bus, etc., were all recognized without any problems.

More complicated symbols often caused that the children were distressed by details and thus did not interpret the symbols according to the intentions.

The kids understood meta-symbols as a car with a red cross was identified as an ambulance.

The children had also problems with abstract symbols as forest, which was interpreted as trees, and town, that was interpreted as houses.

All symbols were interpreted correctly after a short introduction of the symbols. After this, the children mastered the symbols perfectly and were able to combine them in an adequate way. The children remembered also which symbols that were available and were able to fetch them quickly. Based on the tests we performed with the children, the symbols were grouped into eight groups: household items, travel and transportation items, food and beverage items, nature items, toys and play items, humans, faces and emotions, and misc. items.

All symbols have a unique textual name, and Symbo emails consist of a set of such names. This enables i.e. parents to send a symbolic email from a text-based email client. Another feature is culture-based symbols, e.g. a house in Norway will not have the same symbol as a house in Nigeria.

5. Constructing software for children

Traditional software engineering frameworks do not focus on usability, as is necessary when constructing software for children. In stead, they often delay interface related tasks to the late stages of the project, and underplay the importance of user involvement and iterative prototyping. In addition, the task-centered model used by all well-known methodologies does not suit the explorative nature of which children uses computers [6].

The last five years have seen an explosion of educational and entertainment CD-ROM titles for children, which has made children's software a viable commercial market. This new grown market has spawned research articles in the field, and there exists a small body of literature. Nevertheless, there are no complete methodologies, or standard development approaches. This is especially true for the Symbo project, which is neither a computer game nor an educational tool used to teach a specific subject to children. As a result it was necessary to compose our own framework for the development. The methods we have used are therefore a combination of standard software engineering principles, children's software research and our own ideas. Based on our studies of children's role in the development process, we wanted to involve them as much as possible, preferably as informants.

We found that the "Usability Engineering Lifecycle" (UEL) [11] addresses all the mentioned weaknesses of traditional software engineering frameworks while providing a coherent and flexible methodology. The latter was particularly important. The UEL defines a set of tasks, and their order, which needs to be performed to facilitate the usability requirements of the software. It also supplies example techniques to perform the different tasks, but these can easily be replaced by alternatives. This allowed us to try some of the techniques developed specifically for children, such as "Contextual Inquiry" and participatory design with the children [4].

6. Requirements

The purpose of establishing user profiles is to identify different user categories and their characteristics. The data from the profiles will be used to set usability priorities. Ideally they should be obtained from a large percentage of the users, for instance through questionnaires. In our case this was not possible due to time shortage and the fact that the target audience had illiteracy as one of their main characteristics. Instead the data was obtained through an interview with a teacher at one of the schools.

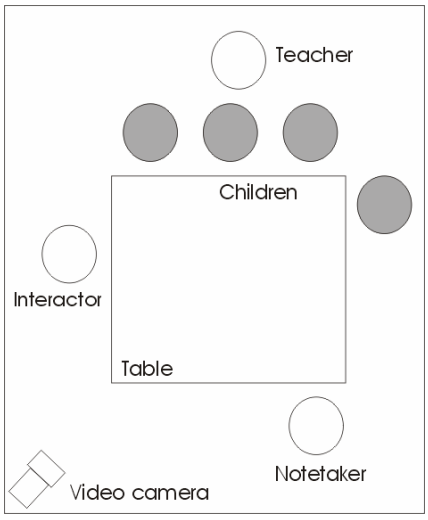


Figure 1: A typical organization of a session

The contextual task analysis replaces the traditional "System Analysis". The purpose is to get a user centered model of how work is performed. This, of course, implies that the task *is* currently being performed, which is not the case in our project. As a consequence we needed to observe *if* they are able to perform the tasks defined, rather than *how* they perform them.

We visited one school and one kindergarten, and had many sessions with the children there. In order to best capture the input from the children we used many of the Contextual Inquiry techniques developed especially for situations where children are involved.

The sessions normally took place in a small study room, and we used either 4-5 children at a time, or pairs of children. To record the events we used a video camera, which we placed discreetly in a corner of the room. Luckily the children seem to quickly forget that they were being filmed. Druin [6] reports that they had difficulties using a video camera when observing children, but we did not experience any of the problems mentioned.

There were three adults present, which performed different roles defined by [6]. One, a member of the project team, functioned as an interactor. His role was to initiate discussions and ask questions concerning the activity. The interactor would not take any notes, rather talk naturally to the children, and become part of their experience. It was important to avoid making the children feel like they were being tested. Therefore the second team member acted as note taker and remained in the background. This approach seemed to work, and we had few problems during the site visits.

There was also a teacher present, which largely remained in the background as well, but offered the children support and reassurance when needed. The placement of the video camera and the people involved can be seen in Figure 1.

Normally one would begin a contextual task analysis by getting the actor (the person who's task is being automated) to perform the task in question. This would not be possible in our case. Firstly, the actors do not already know of, or master the task. Secondly, we did not know whether or not they would be able to do so either. Thus our experimentation needed to start at the most basic level. Our basis was a paper card version we made out of the first draft of the language. This was done simply by printing out the symbols and adding contact sheet paper to make them less flimsy.

We started with symbol recognition. Naturally it would be important that the children understood, and not the mention, had a *common* understanding of what the symbols represented. To test this we placed one card at a time on the table and got them to name the object or symbol. We did this both with just one child at a time naming the symbol or as a group. Figure 2 illustrates a session with four children were we tested the initial symbols.

Figure 2: Symbol recognition

The closest we could get to the tasks in our email client was to get the children to simulate them with the card version of the language. Again, it would not be possible to get the children to perform these based on their current knowledge, so we divided the tasks into several subtasks. The first was to put simple sentences together. This was done by putting all the cards on the table and asked them to tell short stories or statements that we gave them. The second was reading of SymboL sentences. We had made cards with 3 or 4 symbols, which communicated a simple sentence, and asked the children to tell us what they thought they meant. At the end of our first round of sessions we took pictures of the children using a digital camera, which we would print out and make into symbol cards for the next session.

In our second session we took one step closer to the actual tasks and asked the children to write a letter using the symbols. We did this by giving them a blank piece of paper and an envelope. We had added double-sided tape to the back of the symbols so that they would stick to the paper once the children had placed them there. We also gave the children coloring crayons to observe whether adding drawing functionality to the client would be useful.

One of our main goals was to obtain a user-centered model on how to organize the symbols. Usability researchers from the Microsoft Usability Team for children's software have reported great success using a sorting technique to organize items, activities, features and tools into categories [9]. We tried both variations they describe. The first was to announce, once they had written a few sentences and the cards were randomly placed on the table, which we needed help to organize them. The reason we gave was that it took to long time to find any given symbol, which they agreed.

The second technique was to put one and one card on the table and ask a pair of children if it belonged to any of the previous cards on the table. In [9] the usability researchers note that this approach works well for children aged 8 and up. However, we had no problems with using it with children between 5 and 6 years old.

After the sessions the project team members went through the video tape to supplement the notes taken by the note taker and review specific situations.

7. Design

To obtain an accurate conceptual model of how children thought of the system we used the participatory design for children methodology as described in [6]. Using this approach we would ask the children to work with us on creating low-tech prototypes of the software. The hope was that we would identify new possibilities that we otherwise would not have thought of. To facilitate this we used thick pieces of A3 paper that the children could draw on using crayons. We also made our own paper prototypes of the software for them to look at.

Unfortunately, we did not get this approach to work. The children were simply unable to understand what they were meant to do at both test sites. The concept of making something on paper that would later become a computer program seems beyond children at this age. This outcome was not unexpected as the methodology defines it target age group to be children between 7 and 10 year old, which is at least two years older than the children in our group. However, based on our positive results so far in the analysis phase, we found it worthwhile to try it out.

When this did not work out we tried a different approach to get a close view of their mental model. We asked them questions, and tried to discuss with them how they thought traditional communication functioned. In other words, how do you send a letter, how do you receive a letter, what happens to the letter in between sending and receiving? We also tried to find out how they thought electronic communication, such as email, functioned. Finally, we tried to get the children to name the artifacts that would be used in the software, such as "symbol", "email/letter", "sender" and "recipient".

This also turned out to be problematic. They have some knowledge of traditional communication, but no specific idea on how it worked. When it comes to electronic communication some of the children had heard about it, but none had any ideas on its purpose and functionality.

8. Implementation

Figure 3 shows the first screen the user sees when he starts the application. The purpose of this screen is to select who the active user is. At the top one can see the status area, which gives the user an overview of the process, and the current state. Beneath this are the buttons to navigate back or forward. Because this is the first screen the previous-button is disabled, and as a user has been selected, the next-button is enabled. The main area of the screen is occupied by the illustration to the left, and the select-user control to the right. In the lower left corner there are two buttons, "Programoppsett" (options) and "Avslutt" (exit). The former brings up the program's configuration, while the latter quits the application.

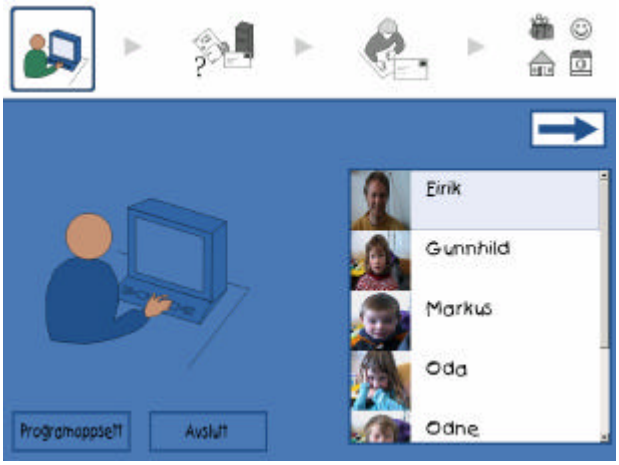


Figure 3: Select active user

The system is designed to differentiate between two types of users, local and external. The definition of the former is users who will be using the client to send and receive mail. In most cases local users are children. External users are people who will send and receive mail to, and from, local users. In other words, people who are in the address book of the children, but who does not use this client. These are mostly adults, but can also be children. One of the reasons for this distinction is to avoid cluttering up the users list in this screen with people who will no be using the client.

Figure 4: Select task (read or write mail)

The current multi-user solution scales well up to one school class. If there are more users it will become difficult to locate people in the list. To avoid this problem if there are several classes using the application, one can simply use multiple copies of the application, one for each class. Naturally, this can easily be improved in future versions, whereby the teacher selects the active class in the configuration.

The next step is to select whether or not the user would like to write or read an email. As seen in Figure 4 symbols are used to visually illustrate the two different choices in the list box.

If the user selected to write an email in the last step, the current step would be to select the recipient of the email. The list box is filled with people the current user has in its friends list (the same as an address book) (Figure 5). The illustration is dynamic in the sense that it puts the picture of selected recipient on the cover of the envelope.

This is the final step and main dialog in the application. Here the user constructs the emails using the symbols from the library. The screen, as can be seen in Figure 6, contains three controls; to the left is a list of groups, at the top is a list of symbols within that group, and in the center of the screen is a "paper control", which holds the contents of the email.

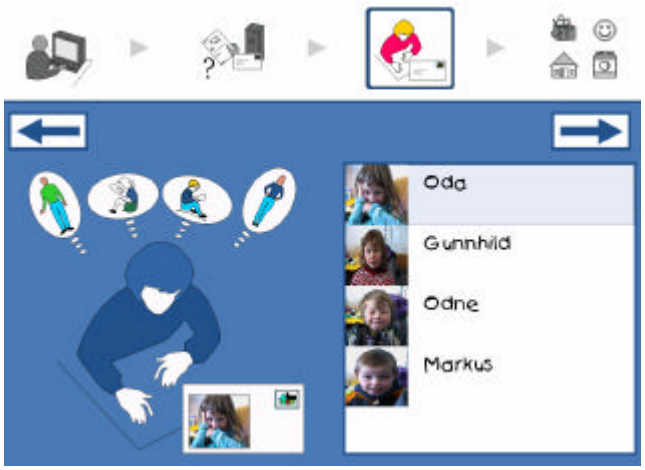


Figure 5: Select recipient

Organization of this screen raises several design issues. Firstly, there are many more symbols in the library than there is room for on the screen. One way to solve this problem is to have scrolling control that contains the symbols. In this way all the symbols would be available to the user without taking up too much screen estate. A better solution would be to do as we have done, and split the symbols into logical groups to decrease the search time. Once the user clicks on one of the categories to the left the top list view will be filled with symbols from that group. The second major issue is how to display the content of the email. Cleary, unlike a normal email client, it would not be possible to use just a standard text control. We therefore need to construct our own control. We know based on our usability goals that we need to provide some sort of structure to writing messages to ensure correct ordering of the symbols. To archive this we developed a "paper control" which had slots that would hold symbols, and arrows indicating the order of the slots.

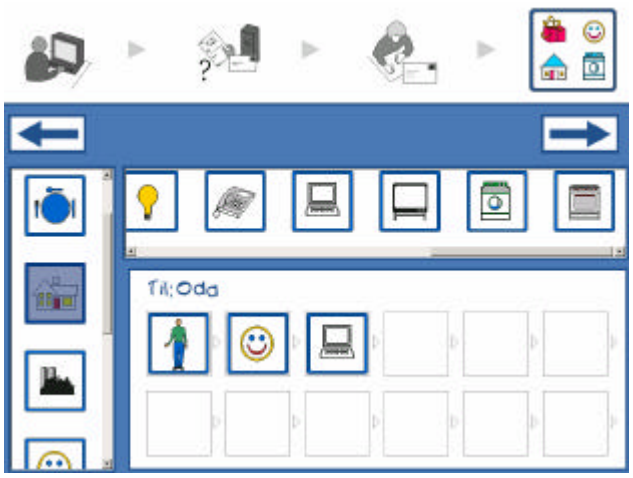


Figure 6: Write email

The third design issue is how to add, delete and reorder symbols. We know from our contextual experiments that the children found the approach of using symbols with double-sided tape on a piece of paper to be very useful. Our goal will be to emulate this, and at the same time take advantage of the opportunities a computerized version offer. We have archived this by using direct manipulation, in the form of advanced and visual drag and drop. As Cooper [3] states, "a visually rich master-and-target drag and drop operation is one of the most powerful operations in the GUI designer's bag of tricks".

The key to successful direct manipulation is rich visual feedback. Cooper [3] defines three requirements to the drag and drop design. The first requirement states that the drop candidate, who in our case is the paper control, must visually indicate its drop-ability. It does this by drawing a thick colored frame around the slot that the mouse is currently

over. In order to separate a drop that would replace a symbol, with one that would simply insert one, the color is either green or red. This can be seen in Figure 7.

Cooper's second requirement is that the cursor must visually indicate that the master object, which in our case is the list of symbols, supports dragging. We accomplish this by changing the cursor from the standard arrow to a hand when the mouse is above the symbol list. The hand symbolizes that we can pick up the symbols. Once the drag operation is started the cursor should reflect this. The optimal solution, which we have used, is to have a transparent image of the object being dragged beneath the cursor.

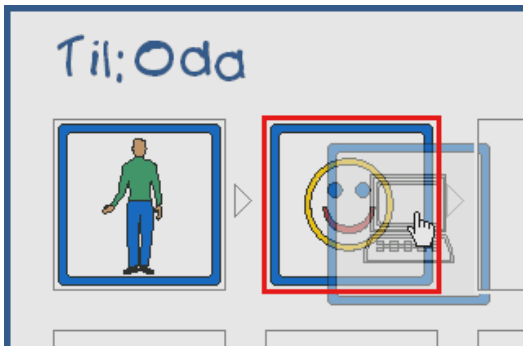


Figure 7: A symbol is dragged above an occupied slot

The third requirement is to give a visual indication that the on completion of the drag and drop operation. In our case this is obvious since the user clearly sees the symbol in its new place on the paper control.

After clicking on the next-button in the write email screen the application will translate the email into text and send it. The client offers visual feedback on this process by showing an envelope that moves from the sender to the recipient. Once the email is sent the next-button is enabled and highlighted. Clicking on it will bring the user to the first screen, whereupon the next child can log in.

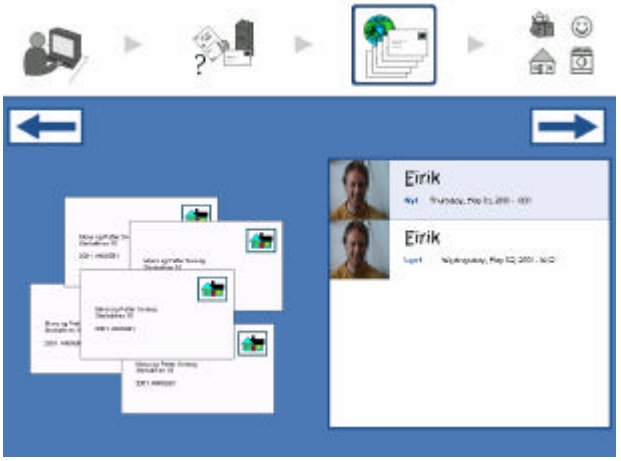


Figure 8: Select mail to be read

If the user selects to read an email (instead of writing one), the client will connect to the server specified in the users profile and download any new emails. Visual feedback on this process is offered in the form of an envelope that moves from the globe to the computer. Once the emails are downloaded the client will automatically move on to the next screen.

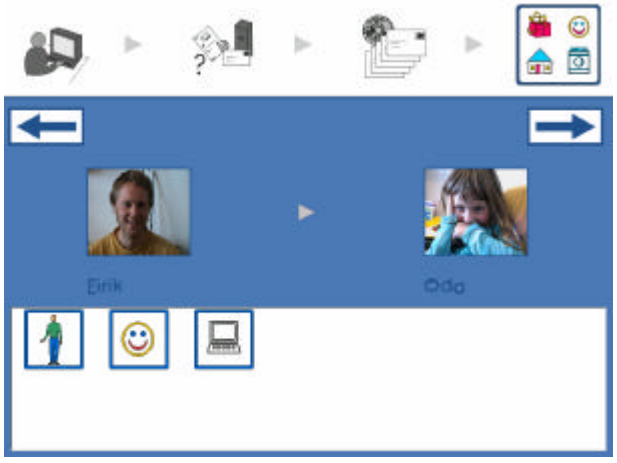


Figure 9: Read mail screen

The purpose of this screen is to let the user select which email he or she would like to read. The list box to the right contains the email in the user's inbox. A picture of the sender is included so that the user can visually identify emails. The list box is sorted by read status, which means that emails that have not been read yet are at the top (Figure 8).

The read email screen presents the email selected in the previous step (Figure 9). Visual cues are offered as

pictures of the sender and recipients with an arrow pointing from sender to recipient. The latter will of course always be the current user. Clicking on the next-button will take the user to the first screen as in the case of writing an email.

9. Results and conclusion

The results received can be divided into four areas, SymboL findings, Symbo findings, usability methodology findings for children, and user interface findings for children.

9.1 SymboL findings

Our major SymboL (symbolic language) findings include:

- The children were able to understand the symbols used in SymboL, and equally important, had a common understanding of what the symbols were.
- Simple symbols worked better than complex symbols. Details either narrowed the definition of the symbol, or confused and made the meaning less common among the children.
- The children were able to understand most meta-symbols, such as red cross, but had problems with abstract symbols, such as buildings meaning "city".
- We were able to organize the symbols into nine groups, identified by the children themselves. These groups are household items, travel and transportation items, food and beverage items, nature items, toys and play items, humans, faces and emotions, and misc. items.
- When constructing sentences in SymboL, children put little emphasis on anything but the object in the sentence.
- Children are able to understand some of the sentences made by adults, and in some cases they are also able to construct sentences that can be understood by adults.
- Not all children in the age range are capable of understanding the concept of using symbols to form sentences.

9.2 Symbo findings

We had three goals for the Symbo e-mail client; ease of learning, personalized satisfaction, and to give the children a sense of communication. Our main results in this category include:

- The software did not contain any major usability flaws, and the children had few or no problems with learning how to use Symbo.
- The children liked the software, and found it both fun and easy to use.
- Most of the children understood what they had done when they sent email, but not how it worked. They enjoyed communicating in this manner.

9.3 Methodology findings for children

In order to take the special needs of the children into account, we had to develop our own approach to software design for small children. The results include:

- Unlike what Druin et al [6] report, we had no difficulties with using a video camera to record the events with 5-7 years old children. The locations were small or medium sized study rooms in a school or kindergarten.
- We did not have any problems using the card sorting approach suggested by Hanna et al [9] for the same age range, even though they reports that the children should be 8 years old or older.

- Compliant with the recommendation in Druin et al [6], we had problems using participatory design and low-tech prototyping with 5-6 years old children (they recommend using children that are between 7 and 10 years old).
- We did not find it necessary to have a teacher or parent present when performing usability testing at the educational institutions, as Hanna et al [8] recommends. This may be because we performed the testing at the educational institution of the child, and not in a usability lab.
- We found the recommendation of Hanna et al [8] to set the cursor speed to a minimum before a child tests the software to be counter productive. The children constantly ran out of desktop space when using the mouse, and they had to be helped by moving the mouse back to the center of the desktop. This may be application specific, as we had few small sized controls.
- We recommend that the size of the scroll bars should be increased from the default setting before commencing with the testing. This makes them easier for the children to operate.
- We found using a prepared script with varying levels of hints, and the smiley scale described by Hanna et al [8] and [9] to be very successful.

9.4 User interface findings for children

Within user interface for children, our results include:

- Drag and drop works just as well for children as for adults. They find it natural, have no problems remembering how it is performed, and enjoy using it.
- When using drag and drop, instead of employing a trash can solution where users can drop unwanted objects, they can drop them where they originally dragged the object from.
- Children aged 5-7 understand traditional communication, and a mail-envelope metaphor can be used successfully.
- Status information needs to be clearer and better presented than for adults in order inform and help the children.
- Illustrations that are used to communicate the purpose of the screen should be backed up with animations, sounds and voice.
- Avoid double clicks and right mouse button functionality. Instead, duplicate the functionality of the left mouse button onto the right.
- Using pictures as visual identification for persons is very efficient, but it may not be a practical solution as they are tiresome to obtain in some cases.

We found that to take the special needs of the children into account one must apply a methodological approach that focuses on usability. It is important to involve the children from the start of the process, and to base design decisions on user interface findings for children, and not assumptions and myths. Based on these findings we constructed an email client. It was evaluated in terms of ease of learning, subjective satisfaction and giving the children a sense of electronic communication.

Acknowledgements

This project would not have been possible without the cooperation and help from Gimle kindergarten and Workinmarka primary School. We would like to thank Eirik Moseid and Gro Nilsen at Gimle and Vidar Lindgård and Anita Hansen at Workinmarka in par-

ticular. The children at Gimle and Workinmarka have all played an important role and deserve credit for all their time and input. We would also like to thank Ellen Dahl at Faculty of Education, Tromsø University College, for valuable comments.

The authors would also like to thank the following persons: Alexander Wilkens for his advice on Microsoft Windows programming; the technical staff at the Department of Computer Science for valuable help with the technical installation; and finally, Tove Midtun, who supplied most of the illustrations used in the e-mail application.

References

1. Bliss, C. *One Writing for One World.* Semantography (Blissymblics) Publications, 1965.
2. Brouwser-Janse, M., Suri, J.F., Yawitz, M., de Vries, G., Fozard, J.L., and Coleman, R. User Interfaces for Young and Old. *Interactions 4,* 2, 34-46.
3. Cooper, A. *About Face – The Essential of User Interface Design*. IDG Books, 1995.
4. Druin, A. Cooperative Inquiry: Developing New Technology for Children with Children. *Proceedings of ACM CHI 99,* ACM Press, 592-599
5. Druin, A. The Role of Children in the Design of New Technology. *Behaviour and Information Technology 21,* 1, 1-25.
6. Druin, A., Bederson, B., Boltman, A., Miura, A., Knotts-Callahan, D., and Platt, M. Children as our technology design partners, A. Druin (Ed.) *The Design of Children's Technology*. Morgan Kaufman, 1999.
7. Druin, A., Stewart, J., Proft, D., Bederson, B., and Hollan, J. KidPad: A Design Collaboration Between Children, Technologists, and Educators, *Proceedings of ACM CHI 97,* ACM Press, 463-470.
8. Hanna, L, Risden, K., and Alexander, K.J. Guidelines for Usability Testing with Children. *Interactions 4,* 5, 9-14.
9. Hanna, L., Risden, K., Czerwinksi, M., and Alexander, K.J. The Role of Usability Research in Designing Children's Computer Products, A. Druin (Ed.) *The Design of Children's Technology*. Morgan Kaufman, 1999.
10. Ingen-Housz, T. The Elephant's Memory – In Search of a Pictorial Language. *Learning Technology Review*, (Spring/Summer 1999), pp. 2-25.
11. Mayhew, D. *The Usability Engineering Lifecycle.* Morgan Kaufman, 1999.