

Innovating with OVID

Daniel Corlett
School of Electronic and Electrical Engineering
University of Birmingham
B15 2TT, UK
djcorlett@iee.org

Object, as opposed to Task-Oriented User Interfaces have now been in use for many years. However, it is only very recently that specific methodologies for their creation have emerged. The aim of the methodologies is to create the ideal, intuitive user interface - the holy grail of HCI. OVID (Object, View and Interaction Design) is just such a method, developed and published by members of IBM's Ease of Use Group [4]. OVID claims to be a 'Complete, correct method' and is the culmination of many years' work including the development of IBM's Common User Access (CUA). The OVID method is followed from project inception to completion, incorporating many familiar interface and software development techniques along the way.

The general principle of an Object Oriented User Interface (OOUI) is to use familiar objects to model the real world as the user normally interacts with it, and then to map the environment onto a computer. OVID has already been used successfully in this context at IBM, designing software such as RealPhone and RealCD.

However, if instead of simply automating everyday tasks on a computer, we decided to innovate and develop new working environments and practices, then how easy is it to use such a modelling technique to create the ideal interface? A group of six student software designers and programmers discovered the answer as they put OVID to use in the creation of HandLeR.

HandLeR (Handheld Learning Resource) is a cross disciplinary project hosted by the University of Birmingham, UK. The intention is to design a ubiquitous computing and communications device to aid and support learning throughout a lifetime. This device should offer all the productivity tools expected of a personal computer in a small, portable package, but more than that, it must act as tutor, guide, mentor, secretary, companion and memory aid. Intuitive to use, it should be available continuously and persistently throughout the user's lifetime, adapting with the user's personal and intellectual development. The team's brief was to develop a first generation functional prototype in 8 months. Given the scope of the project however, it was decided to narrow the study to children 7 to 11 years old [2].

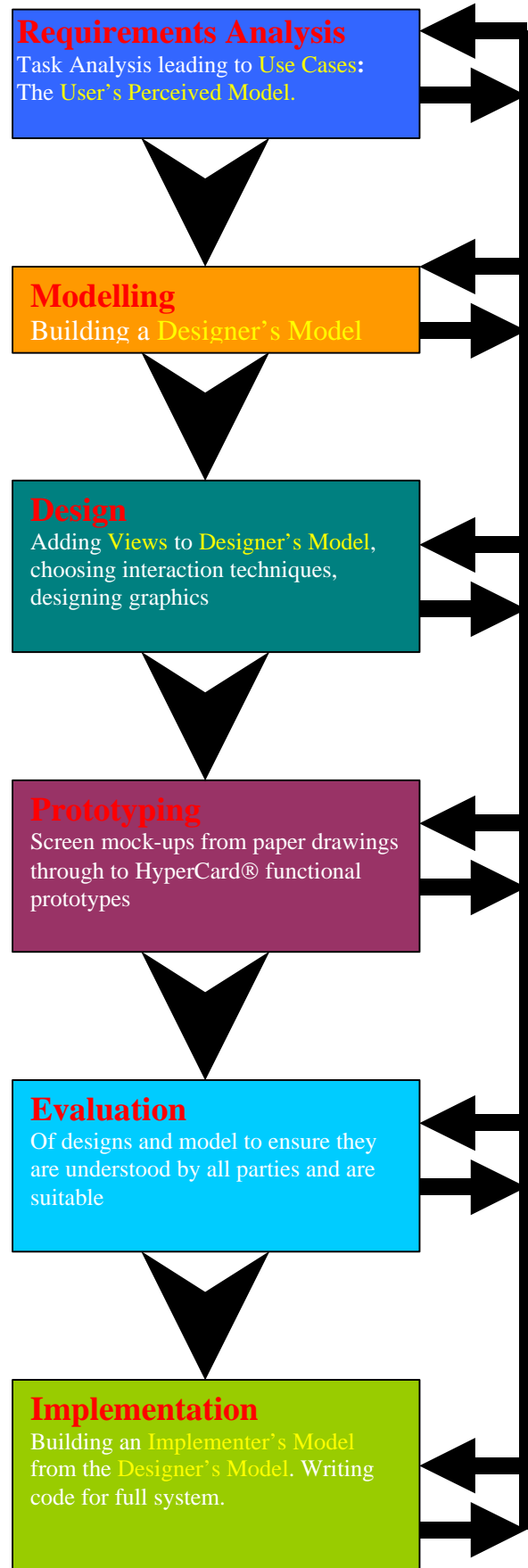
The team was convinced early on, that in design, they should break with existing interface and metaphor conventions to provide something which would be intuitive to all users and appropriate in every context of working and learning. The first three months of design had already passed before using OVID was suggested. This time was spent in brainstorming the concept and conducting user surveys to define a generic HandLeR, out of which came the *space of possible designs* - a collection of story-boards, sketches and outline specifications, covering the interface as well as the technical and ergonomic aspects. As decisions were made, and system-specifics began to arise, it was decided to formalize the design with OVID.

The main steps to designing with OVID are shown in the diagram [figure 1]. Owing to the iterative nature of OVID, most or all of these steps may be revisited at any time. Thus, OVID is involved in a project from inception to completion and involves all members of a software team. Learning OVID presented a steep learning curve to the group, not least because the skills of Object Oriented Design (OOD), Unified Modelling Language (UML), Task Analysis and Evaluation had to be learnt at the same time. Simply learning how to do Object-Oriented modelling can take up to six months [7].

OVID models use a subset of UML, and Rational Rose®, a Computer Aided Software Engineering (CASE) tool was used in the project, though it does not yet support all the necessary notation for OVID. The model is built incrementally, starting with the *user's perceived model* of his environment. Out of this develops the *designer's model* - a collection *objects* key to the user's task, to which *views* are associated. Interaction diagrams, state diagrams and state tables ensure completeness of the design, and finally lead into the *implementer's model*.

Figure 1: Flow Diagram of the OVID method.

A “complete, correct method” from Requirements Analysis to Implementation.



Requirements Analysis and the User's Perceived Model

This was the longest stage in the HandLeR project lasting several months. It incorporated a thorough analysis of the users' needs and current situation. In addition, creative thinking was applied to see how the users' tasks could be improved by a HandLeR.

The input to an OVID design comes from the designer observing or assuming the role of the user and describes in natural language, the actions that they perform and the objects with which they interact. These descriptions are *use cases*. Observation, rather than lateral thinking is key. This became the greatest conflict between the method and what we were trying to achieve. In attempting to make the system innovative and generic for all users, we had already decided to dispense with workplace-specific objects such as desktops, folders, mailboxes and so on.

Prior to using OVID to guide the design, we had already decided on a main facet of the interface which had come solely from brainstorming the problem. We tried to consider what would be the most intuitive metaphor for the system, and in doing so decided on the human body. Every person understands the functions of their own body – the legs for transport, the hands for manipulating things, the eyes for seeing, etc. Most importantly we know the brain is for thinking and remembering. So we decided to base all functionality of the system on the human body and provide the user with an avatar, or character to represent all the fundamental actions they may wish to perform. For example, clicking on the eyes of the avatar would access the built-in camera, clicking on the mouth would begin a phone conversation (through the built-in mobile telephone) and clicking on his head would open the mindmap. The mindmap dynamically links all the user's experiences, from web-browsing to writing a document, to pictures taken with the camera and more. All links between objects are made by context associations such as key words, locations (provided by the integrated Satellite Positioning System), time, and others. We are confident, that had we left the design entirely to 'mimicking' the existing real-world, this design would not have come about. However, we did use OVID to refine the design ideas. In this HandLeR designed for children, the avatar manifests itself as a cartoon rabbit [figure 2].

It was decided first to model the system tools (for drawing, writing, communicating etc.), which would be based on recognizable implements such as books, pens, erasers, paintbrushes etc. Task descriptions are fundamental in making a successful model, but writing them can be tricky. Sentences like "*Child writes in topic book*", "*pen writes in topic book*", and "*Child pushes pen across topic book*" sound alike, but the resulting object model could be considerably different. The necessary level of detail also became a contentious issue. Is "*Child paints picture*" at all helpful? Descriptions like "*Child opens hand to grasp paintbrush. Paintbrush is dipped in water and paint. Paintbrush is pushed across paper to leave a mark.*" would certainly ensure that the project was never completed! We realised that most importantly, descriptions should be consistent, and if serious doubts were raised, then the users should be revisited to better understand their perceptions. In practice, descriptions remained brief and informal as the team were all involved in all development processes and fully understood what was being implemented. Working as a small team in the same office allowed any questions to be answered immediately. However, with larger projects and more team members, it is likely that more comprehensive descriptions would be required. Examples of descriptions are found in [figure 3].

Next, we tried to model the novel parts of the system. This created something of a paradox. In discussion with one of OVID's creators, we decided to 'imagine' that HandLeR already existed, and write the task descriptions as if we had observed someone using it. Since we were writing the mindmap heuristics from scratch, the process of imagining someone using it helped us to refine our decisions and make a more intuitive navigation system. HandLeR is not only a passive system, but is also intended to include autonomous agents which had to be part of the system model. To this end, we took the liberty of defining HandLeR as a user of itself, an odd concept, but one which worked.

For unknown tasks and objects, it would have been appropriate to start with *Essential Use Cases* [1]. *Essential Use Cases* explain abstract intentions or goals without task-specifics. These can then be elaborated through the imagination of user and designer to generate more full *use cases*. However, it is then important to refer any objects that have been 'conjured up' back to the user to test for intuitiveness. Not doing so was a mistake that cost dearly later in the evaluation cycle.

The set of *use case* descriptions were rewritten many times before a consistent and informative set emerged. OVID being an iterative method allowed the descriptions to be revisited at any stage, and it was often the later stages, particularly interaction and state diagrams that showed when descriptions had either too much or too little detail.

The next stage was to extract and sort all the nouns that occur in the *use cases*. They are placed in a

table under *concrete objects*, *people (objects)*, *people (subjects)*, *forms* and *abstract objects*. These are then ordered in terms of importance and frequency of use. Some people can be both object and subject, as in “Child gives completed homework to the teacher” and “Teacher gives marked homework to child”. It is important to realise who is using the system at the time so as to maintain consistency and avoid confusion. Some items were difficult to classify – is a picture a concrete or abstract object. The paper or canvas may be concrete, but is the image itself? At this point it was necessary to return to the user’s point of view to see how they would normally associate with such an item. As a rule of thumb, if in some form it would be possible to touch it, then it was *concrete*, if not, then it was *abstract*. Thus an image was considered *concrete*, whilst a conversation was labelled *abstract*.

Once all the objects are identified, each must be described in a single clause sentence. This helps to define membership and behaviour of the *classes* (objects) in the diagrams. The rule of thumb is that if a description contains any ands, ifs, or buts, then it cannot be used consistently in a user interface and probably needs to be split into further objects. [4]

A metaphor should normally only be used if the task analysis makes it apparent [1]. The final set of descriptions did not directly support the chosen metaphor, but the intuitiveness of the human body as the place where all the user’s tasks begin was borne out through user studies and was retained.

Modelling – the Designer’s Model

This stage was shorter than the first, but nevertheless took many weeks to build an entire first generation model, and then a few days each time the stage was revisited to update the model.

Having defined the objects, it was time to start building the *designer’s model*. First, objects were added to the UML diagram as hierarchical *families* of classes [figure 4]. At first, the designers made the mistake of classifying objects according to use rather than behaviour. It seemed natural to place together paint, brush, and crayon as drawing tools, despite their individual properties being totally dissimilar. Likewise, pens, pencils and erasers were grouped as writing tools, when actually an eraser performs the opposite action to the other two. A further iteration of object tables and descriptions was thus necessary. ‘Cunning’ definitions were employed. Paint was removed and a paintbrush was assumed to be already loaded with paint. This was then grouped as a marking tool, together with pen, pencil, crayon **and** eraser - ‘...creates a mark the same colour as the background’.

Once happy that the objects were suitably grouped, new diagrams were drawn with all of the other interrelationships of objects. These are then augmented (very easily) to include the *views* for each visible (or audible) object [figure 5]. A view can constitute anything from a single button or sound to an entire screen.

Interaction diagrams, (still part of the *designer’s model*) came next [figure 6]. These illustrate *use cases* for the **new** system. Interactions between the user and views are detailed, as is the information that must be passed internally between views. It was at this point that we realised we had not defined the behaviour of the mindmap nearly well enough. Many iterations were rapidly cycled. A basic diagram was drawn, and was tested by performing a mental walkthrough of the *use cases*. This would highlight the errors, omissions and inconsistencies, and the diagram would be redrawn. Eventually the diagram became comprehensive and consistent enough to satisfy the evaluations.

Finally, state diagrams were drawn to show how objects change state for given actions [figure 7]. The diagramming process ensured that all permutations were considered. However, even at this late stage, inconsistencies in the model were uncovered. Previously, the designers had considered that for example, a piece of paper, initially blank, could ‘become’ either a picture or a piece of text. However, such object *reincarnation* is not possible in state diagrams. Objects may change state, but they cannot become other objects. It was realised that not only could the diagrams not be drawn, but such a notion would most likely confuse the user. So, definitions were redrawn. A generic workbook was introduced and defined as a container, whilst paper and exercise books were removed. Objects such as text and images could then be added to or removed from it, each only beginning and ending life once.

At an early stage, nouns had been extracted from the task analysis *use cases*, whilst all other information was discarded. However, when describing the relationships and responsibilities of objects, particularly at implementer’s level this is not enough. Points of View (POVs) [5] are very useful for this. A POV (which can be written in natural language like all other OVID descriptions) not only describes each object’s attributes, but the rest of the model as seen by that object. A Rigorous application of such descriptions can eliminate exceptions and omissions in functionality.

Although OVID uses task analysis to extract objects, the method is not inherently task-oriented. Were the prototype HandLeR to have developed further to include more custom learning and creativity tools, this could have been very limiting. Working practices are not all step-by-step procedures with clear objectives and set tools. For example, each person uses a calendar or diary in a different way [8]. Often, entries are uncertainties without crisp formalities. The objects in this scenario may be quite simple to identify, whilst how they are used may require a detailed task and organisation study.

Design

This stage occurred partly in parallel with the previous one, lasting as long, but requiring only a few hours per week. As sections of the model were completed, graphic design could begin.

OVID does not explicitly explain how real views are to be developed from the abstract view classes in the diagrams. One of the hardest decisions to be made was how to embed views within each other, such as buttons or text boxes within screens. The interaction diagram was a good guide, but a more thorough approach would have been to create comprehensive POVs for views as well as objects. In practice, the group used a combination of intuition and trial and error.

A means of bridging the gap between models and design is still needed. In hindsight, the later parts of the modelling stage and the earlier parts of the design stage should have been merged. Content modelling [1] using the abstract views on sticky notes can lead to logical collections of views into 'super-views'. For specific use cases, a whitewater approach [3] with user participation can rapidly build up interface views (still devoid of graphics and specific interaction techniques). This defines the placement of views within other views, and may identify views not yet explicitly shown on the diagram. Content models can then inform augmentation of the original view diagram with additional views, relationships and hierarchies. This more complete diagram, with reference to the content model too, can in turn inform the creation of useful interaction diagrams, which more closely relate to the finished system. As the interaction diagrams are built up, *navigation maps* [1] can be used to test for usability of a given network of views. This would complete the modelling stage (for at least the first iteration) and graphic design could ensue.

As the designs grew, the previously separate metaphor and tools began to merge. When the avatar was drawn, it was noticed that the hands were valuable parts of the body, but were not being used. It was also clear from the *use cases* that writing and drawing were important tasks, and ones which had to be clearly represented. So, we chose to place a workbook in the one hand and a palette in the other [figure 2]. Touching these would launch the appropriate tools. Having now placed these objects in the avatar's hands, it was necessary to redraw the models to relate the objects and define the new interactions.

Prototyping

Various methods of prototyping were employed, ranging from low-fidelity paper prototypes through to semi-functional HyperCard® screens. The advantage of paper prototypes is that many can be quickly produced, and several iterations of design were drawn before any evaluations were carried out with the children. The more elaborate HyperCard mock-ups were later modified and recycled as tutorials for new users.

Evaluation

Each evaluation was completed in a full day, and usually one or two days were spent afterwards inspecting the results and figuring out how they should affect the design. Paper mock-ups were placed in front of the users for evaluation. The main attribute tested was intuitiveness – the children were given hypothetical tasks and were asked from the screen designs, what they thought they would have to do to perform the task. Had there been more time, longer cognitive walkthroughs would have been conducted at the low-fidelity stage to ensure that whole sequences were easy to perform. Instead, the designers performed these mentally on the interaction diagrams.

The mock-ups were then enhanced via the *implementer's model* to fully-functional Visual C++ code, which in its early stages was hi-fidelity mock-up, also to be evaluated.

At each stage, an evaluation could force a change in the models which would then need to be reworked. As the models grew in size, it became increasingly hard to track changes and their effects throughout the model.

Implementation

The *implementer's model*, built on the *designer's model* was then drawn up. Extra information needed by the programmer is included and any functionality not directly connected with the interface must be

defined, such as the database behind the mindmap. In addition, since in the prototype HandLeR some proprietary tools were used, the interactions with these, rather than the ideal tools had to be modelled. State tables also had to take account of the fact that a computer sees objects differently from the user [figure 8]. For example, a piece of text, either '*existent*' or '*not existent*' to the user, is to the computer a file which can be '*open*', '*saved*', or '*modified*'. Building the implementer's model generally took only a couple of days each time a generation of the designer's model was completed. Coding the implementation took the equivalent of three people full-time for two months.

Finally...

Final evaluations ensued, consisting of cognitive walkthroughs and heuristic evaluations. Children were given a HandLeR to use for a number of hours. For some of this time, they had specific tasks to perform, the rest was spent in informal exploration. During this time they were filmed, and afterwards, interviewed and given questionnaires. Many more children were given printed versions of the interfaces with questionnaires to test intuitiveness.

The main concern involving the model was that only 10% of the children found the feet of the avatar an intuitive place to start browsing the web and only 7% associated the watch carried by the rabbit with a diary and journal, which showed that earlier evaluations may not have been thorough enough. However, all other aspects, including navigation of the mindmap, drawing, taking photos and making a phone call were found to have surprisingly high success rates. Some were recognised at first glance by as many as 94% of the children. Clearly applying the thorough processes of OVID had paid off.

Nearly all of the children expressed a dislike for the aesthetics of the cartoon rabbit, but each had his or her own ideas as to how it should be improved. Clearly, the avatar should be highly customisable, maybe taking on the personas of film and sports stars, aliens, robots or teddy-bears.

OVID and its UML diagrams are ideal for team collaboration and *informant design* [6] Until the implementer's model is reached, all models and descriptions are written in the user's familiar language at a level basic enough for all parties to understand. This allows even young children to participate in the design. The implementer's model is simply an augmentation of the designer's model, this allows either designer or programmer to quickly understand the other's model and to build one from the other with minimum additional work. The object-oriented approach allows the design to be directly transferred to code, which is supported by the case tool. What would be useful, however, is a CASE tool that supports team work and versioning, and also the attachment of images to views to be shared by the whole group.

Had OVID been employed from the beginning, it is unlikely that the (very successful) body metaphor would have become apparent, nor would the mindmap have been considered. In an attempt to avoid the files and folders metaphor, a shallow task analysis of the situation would have seen children using drawers, display boards, cupboards and marking piles for storing and retrieving information. Whilst these might have been ideal for a system only used in this context, they could not expand to support *lifelong learning*.

The study proved that OVID can be used successfully for a project like HandLeR. However, it must be applied with careful consideration. It is necessary at each stage to decide when to apply OVID rigorously, and when to ignore the constraints imposed by it. The evaluations supported our experience that good graphic design is not guaranteed by an OVID model, but that excellent interactivity could be.

Above all, before even task analysis is conducted, it is essential for the designers to fully understand the goals and requirements of the project, and be prepared to hold off using OVID whilst some initial creative problem-solving is carried out.

1. Constantine, L.L. and Lockwood, L. *Software for Use: A practical Guide to the Models and Methods of Usage-Centered Design*. Addison Wesley, 1999.
2. Corlett, D.J. *et al.* Group Design Study. Final Report: Vol. I - Development of the Interim HandLeR. 1999. Available from School of Electronic and Electrical Engineering, University of Birmingham. B15 2TT. UK
3. Dayton, T. *et al.* Bridging User Needs to Object Oriented GUI Prototype via Task Object Design, in Wood, L.E. *User Interface Design*. CRC Press, 1998
4. Roberts, D. *et al.* Designing for the User with OVID: *Bridging User Interface Design and Software Engineering*. Macmillan Technical Publishing, 1998.
5. Rosson, M. B. Designing Object-Oriented User Interfaces from Usage Scenarios. Available at <http://www.cutsys.com/CHI97/Rosson.html>
6. Scaife, M. Rogers, Y. Aldrich, F. *et al.* Designing For or Designing With? Informant Design for Interactive Learning Environments. CHI'97: Proceedings of Human Factors in Computing Systems, 1997.
7. Van Harmelen, M. In Proc EG Workshop on Design, Specification and Verification of Interactive Systems, Springer-Verlag, 1996, pp199-231.
8. Van Harmelen, M. *et al.* Object models in User Interface Design: CHI 97 Workshop Summary. Available at <http://www.cutsys.com/CHI97/Results.html>

Figure 2: The Rabbit Avatar

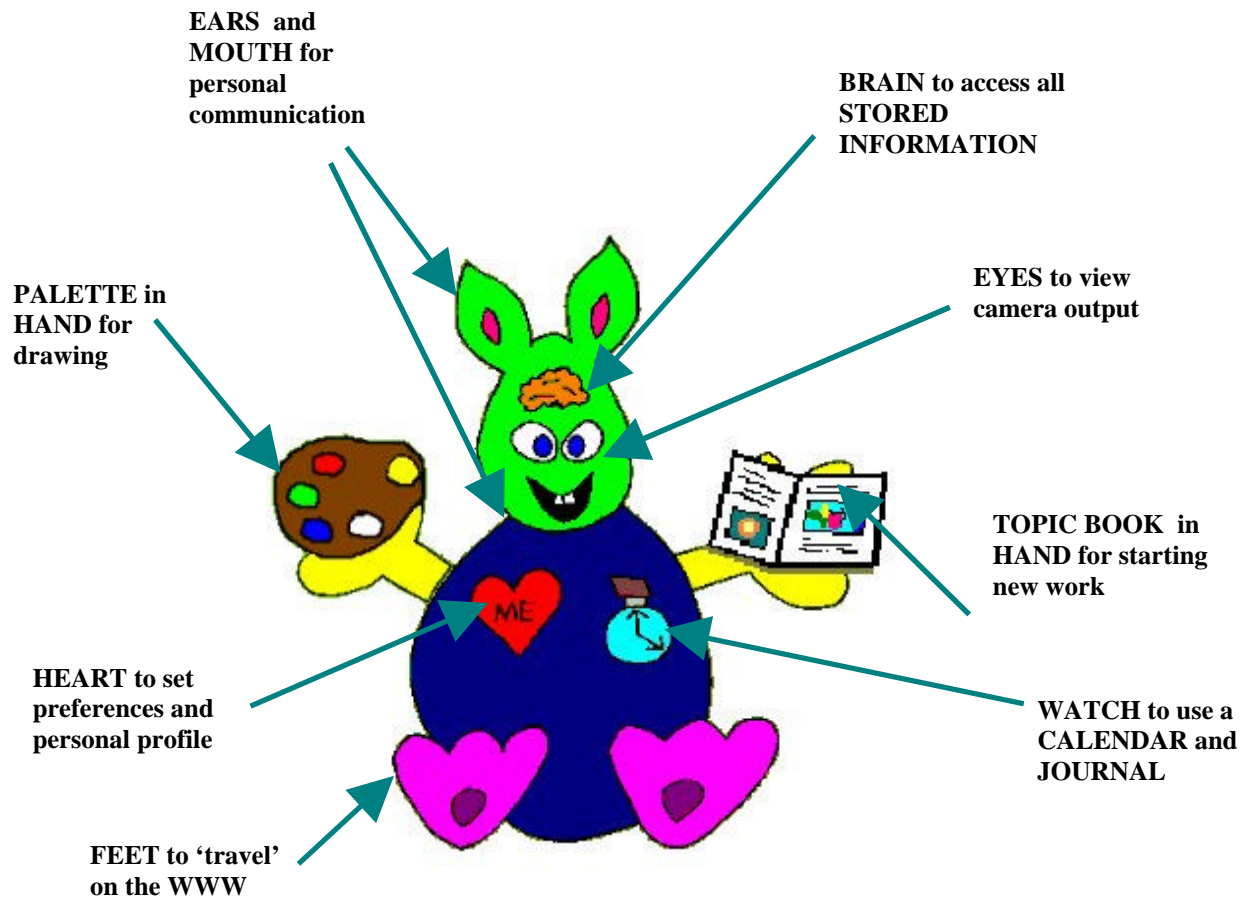


Figure 3: Use-Case descriptions and noun extraction

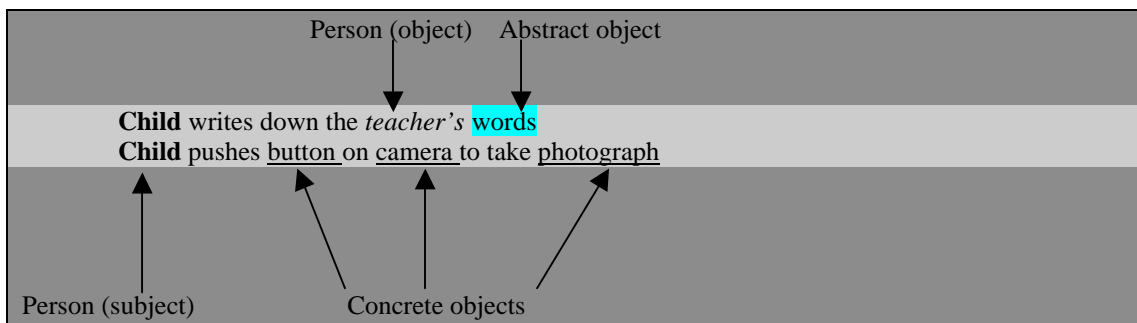


Figure 4: Object hierarchy of parts of the avatar

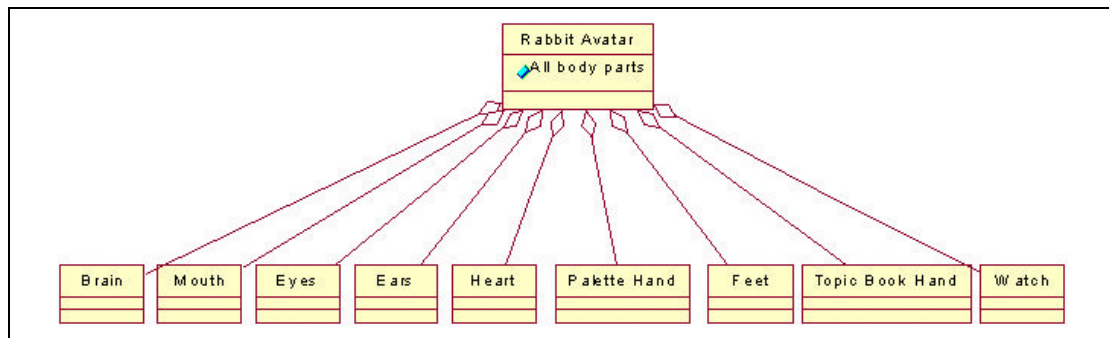


Figure 5: Object Relationships Diagram for avatar, augmented with Views

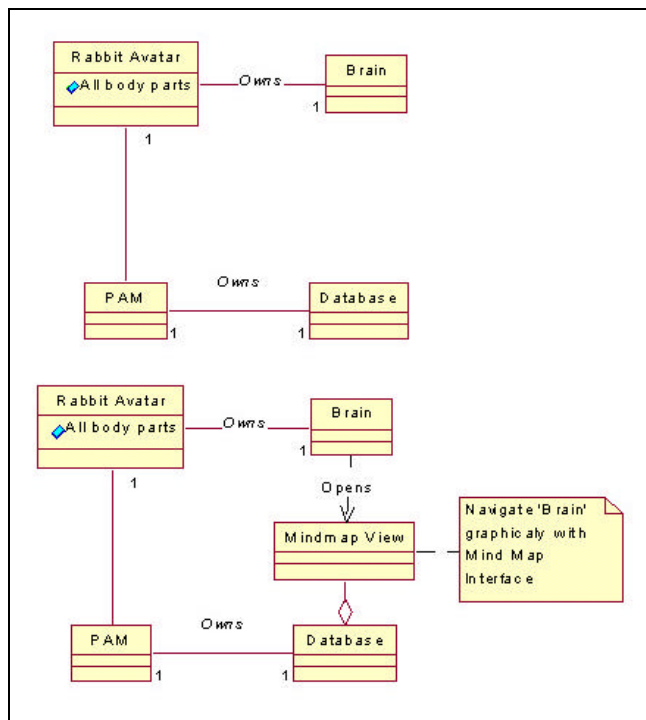


Figure 6: Interaction diagram for avatar view

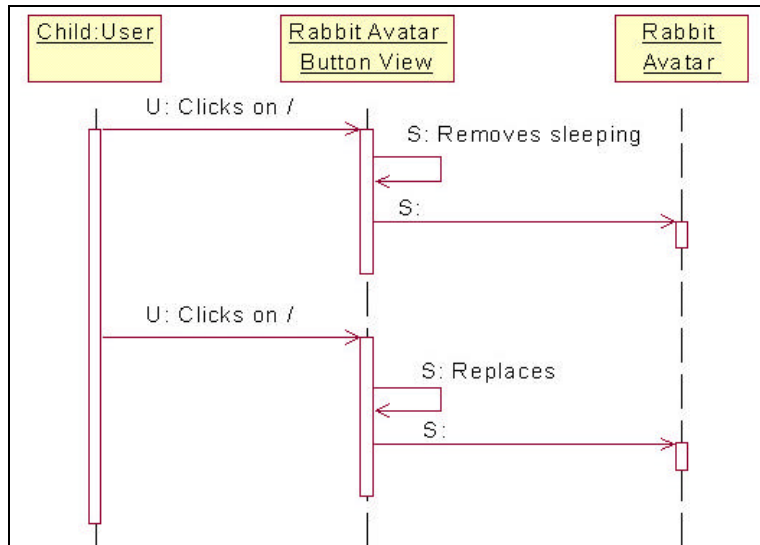


Figure 7: State (Harel) diagram for topic book

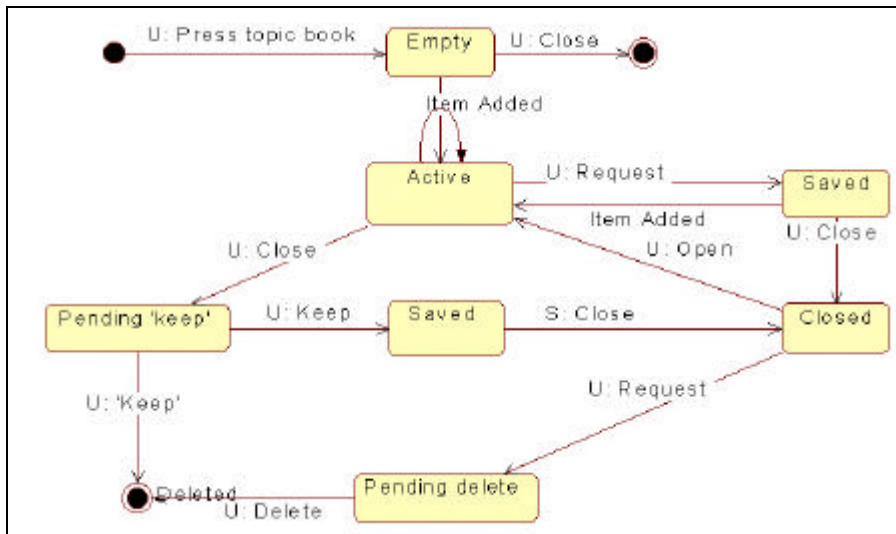
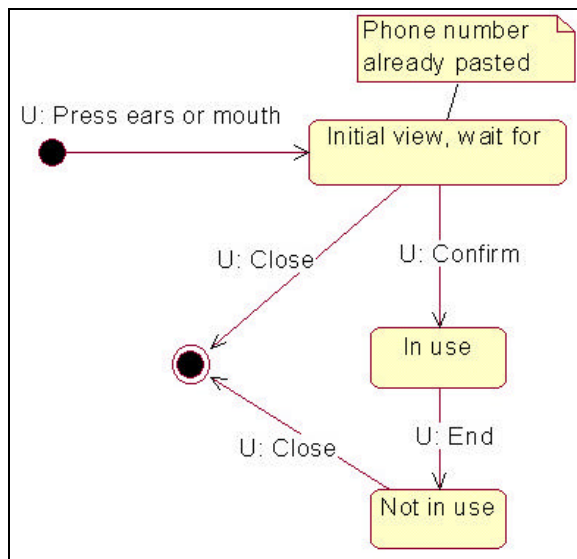


Figure 9: State diagram for proprietary phone software (from implementer's model)





A HandLeR device in use