mCAT - a Method for Co-ordination Across Technologies

Kari Hamnes & Silja Nyhus
Project I
Telenor R&D
Instituttveien 23
N-2027 Kjeller
Norway
+47 63 84 84 00
kari.hamnes@telenor.com
silja-ingeborg.nyhus@telenor.com

ABSTRACT

In this paper, we discuss the general problem of developing user interfaces in a multiple-UI paradigm, accentuated by the latest developments in mobile and handheld interaction and new interaction technologies. The review section discusses the specific problem of consistency across alternative user interfaces (using different interaction technologies) for the same application. Having identified the need for a method for co-ordinating user interfaces across interaction technologies, a set of interviews and a case study are used to identify further requirements and to derive a first-pass evaluation method. The mCAT method (method for Co-ordination Across Technologies) uses consistency analysis as a tool for co-ordinating alternative user interfaces across interaction technologies. Finally, the paper discusses preliminary results and outlines future work to evaluate and further develop the mCAT method.

Keywords

Multiple-UI development, handheld interaction, mobile devices, PDA, consistency

1. INTRODUCTION

Current developments in mobile and handheld computing have broadened the spectrum of interaction technologies that are now being used to implement user interfaces.

Whereas traditional computers generally use a keyboard, a mouse and a graphical display to support interaction (GUI - Graphical User Interface), handheld and mobile devices may include keyboard and pen-based input, character recognition and even limited speech recognition. Handheld devices (or PDAs - Personal Digital Assistants) like Palm™ and Psion™ are by their nature small, and have limited screen size, resolution and number of colours. Communication devices like mobile telephones and WAP (Wireless Application Protocol) telephones have even more severe limitations, both with respect to input and output. In spite of the limitations, these mobile and handheld devices are becoming increasingly popular because of their usefulness and entertainment value for people on the move.

In this setting, telecom operators like Telenor are attempting to offer services that answer to the requirements of 'any time, any place, any device' (in Microsoft terminology). However, in spite of the potential usefulness of such services, the situation may not be only good for end-users, and may be a lot more complicated for developers.

Traditional development has tended towards monolithic development for each technological platform, and

consequently for each interaction technology. Specialists on telephone based interfaces have worked on telephony services, and specialists on mobile telephone services have worked on mobile telephony services. This setting can be called a single-UI (User Interface) paradigm.

In the current situation, the functionality is increasingly independent of the interaction technologies. A set of functions (e.g. e-mail) may have several different user interfaces, implemented using different interaction technologies (e.g. traditional e-mail, e-mail delivered via a web interface, e-mail delivered as synthesised speech over the telephone and e-mail delivered as SMS messages). The different user interfaces cater for the different types of devices that end users may use to access the services, either when they are stationary or on the move. We will call this a *multiple-UI paradigm*.

From end-users' point of view, the multiple-UI situation is not dramatically different as long as they use only one type of interaction technology to access services. However, many users use several different devices, e.g. a PC, a telephone/mobile telephone and a PDA, and may potentially have to interact with three or four different variants of each service.

Developers of such services have an even more unenviable situation in moving from a single-UI to a multiple-UI paradigm. From a functional point of view, a service must in fact work for the different interaction technologies. At the same time, an acceptable (to the end user) level of usability must be achieved for users of each variant, and also for users that use more than one variant. This requires the co-ordination of development for different interaction technologies with the aim to achieve usability. This paper discusses a partial solution to the problem of co-ordinating multiple-UI design, while trying to ensure an acceptable level of usability.

2. CONSISTENCY

2.1 Why Consistency?

Consistency is generally accepted as a design goal for interactive systems, and one that contributes toward increased levels of usability (Nielsen 1989). In (Shneiderman 1992), "Strive for consistency" is listed as the first rule among eight golden rules for user interface design. However, the benefits of consistency are contentious, something that the discussion between Grudin and Wiecha et al. confirms (Grudin 1989; Grudin 1992; Wiecha et al. 1990; Wiecha 1992).

In (Caulton and Dye 1997), a study of consistency is reported and the claim is that for vertical applications, where the user's goals differ for each application, consistency is less important than how well the application supports the user's goals. For applications that have many similar functions, consistency may have a better effect.

Potential end-user benefits of consistency may be increased transfer of skills from one system to another and fewer errors (Nielsen 1989). Nielsen also lists potential benefits for the user company and the vendor company due to lower training costs, less demand for customer support and increased sales. In (Reisner 1993), the argument for consistency is that it allows users to generalise and make inferences that help in learning and using the interface.

The discussion and disagreement about the merits of consistency seem to stem in part from a lack of definitions. In (Kellogg 1989), Kellogg feels that consistency has no meaning on its own, as it is inherently a relational concept that must be defined as such. In (Reisner 1993), there is an attempt to remedy the situation by providing a formal description of inconsistency. Reisner derived the following two-part definition of consistency: “doing similar things in similar ways” AND “agreement between agents about which things are similar”. This definition (at a high-level) takes into account Kellogg's statement and identifies different viewpoints (e.g. a user, a developer) for what might be “similar things” and “similar ways”.

2.2 Dimensions of Consistency

In order to define consistency in more detail for the purpose of our research, we explore the notion of consistency dimensions. Dimensions of consistency are dimensions along which consistency may be achieved.

In (Nielsen 1989), the levels of consistency are discussed according to where the consistency applies, e.g. for an individual application, across a product family etc. This is a more fine-grained division of application context than provided in (Grudin 1989), namely that of internal vs. external consistency. For our purposes, it seems sufficient to discriminate between internal and external consistency.

When trying to achieve consistency across different interaction technologies, we used Nielsen's protocol model to identify and analyse design elements that were seemingly independent of interaction technologies (Nielsen 1986). These elements have a potential for being consistent across interaction technologies.

In (Grudin 1989), metaphor (analogy) is discussed as a dimension of consistency, i.e. consistency between real-world and system concepts or objects. Metaphors are categorised as a sub-dimension of external consistency in (Kellogg 1989), but for the purpose of our work we will use metaphors as a separate dimension for the analysis of consistency. Metaphors are high-level design elements according to the protocol model in (Nielsen 1986).

The sequences of operations, or dialogue, is another dimension that we will explore as a dimension of consistency: it is closely related to task, it can be made independent of interaction technology and procedural consistency is often stressed in design guidelines (Reisner 1981). Sequence is categorised at the syntax level in Nielsen's model (Nielsen 1986).

The names and labels of functions and objects in the system, or vocabulary, is classified as a lower level dimension (Kellogg 1989; Nielsen 1986), but is still to a degree technology independent. From now on we will call this dimension vocabulary.

Finally, an interesting dimension (from a commercial point of view) is the extent to which company or brand identity can be made consistent across different interaction technologies.

2.3 The Need for a Method

Although there are development strategies for co-ordinating user interfaces across platforms and products (e.g. as reported in (Nielsen 1989)), these strategies focus on co-ordination across slight variations of the same type of interaction technology (e.g. across GUI-style interfaces).

There are tool developments that aim to promote and ensure consistency (e.g. Löwgren and Lauren 1993, Vanderdonckt and Bodart 1993; Wiecha et al. 1990). These tools also address limited variations of specific interaction technologies (GUI-type interfaces). In (Nielsen 1989), the task of achieving consistency across interaction technologies is considered to be particularly difficult. However, with the current move towards a multiple-UI development paradigm, using many different interaction technologies, attempts must be made to address this problem. In this paper we propose a method which addresses this problem specifically, and attempts to provide a partial solution to it.

2.4 Requirements Derived from Interviews

In order to develop a method that addresses the actual real world concerns of developers, we interviewed four user interface developers with HCI and usability background. All four developers work in product development for multiple-UI services. The interview subjects are representative of the target users of the method to be developed.

The experts were specialists in the design and evaluation of systems using different types of interaction technologies that are relevant for the scenario described in the introduction:

- Interactive Voice Response (IVR)
- Speech technology (recognition/synthesis)
- Speech synthesis combined with IVR
- WAP style interaction
- web-style interaction on PCs and PDAs

The interviews were semi-structured, and included questions on general problems of developing multiple-UI applications as well as on problems of achieving consistency across interaction technologies. The relative importance of the consistency dimensions identified in our review (metaphor, sequence, vocabulary and brand identity) and problems of achieving consistency with respect to these dimensions was also covered in the interviews. Each session lasted between 1 and 1 1/2 hour, and was recorded and later transcribed (verbal protocol).

The experts raised interesting issues and problems that should be taken into account in the development of the method. The interviews were all performed in Norwegian, and we have translated and re-formulated the main points as requirements below.

The primary consideration should be that the user interface supports the user's goal and task, i.e. the main tasks are of paramount importance.

The type of user and context of use should be used to determine criteria for co-ordinating user interfaces.

One may have to accept inconsistencies in order to exploit the strengths of each interaction technology, and should not design sub-optimally in order to achieve consistency.

It is more important to achieve consistency across alternative user interfaces for frequent tasks than for infrequent tasks.

The method could include the development of a definition list for concepts and terms (vocabulary) that is used across the user interfaces to be co-ordinated.

Product development cycles are often short, and the requirements for effectiveness on the part of a method is of the essence. The method must show immediate utility and not require extensive training (e.g. should include simple checklists or templates, and be configurable to the type of product development cycle).

Limitations in the interaction technologies (e.g. recognition errors, display limitation) may influence consistency with respect to vocabulary. One should balance the demands on the user and the system in order not to induce errors (e.g. consistent vocabulary may ease learning, but may increase recognition errors and subsequently lead to task errors or failure).

Interaction technologies may have well-established uses, and when applied in new contexts may influence user expectations and perception of metaphors and objects in the system. For example, when making a call to have your e-mail read out to you, the call itself may be perceived as an object, whereas in an e-mail client on a local area network with continuous Internet access, the 'call' object does not exist.

There are problems that may preclude consistency if the strengths of new interaction technology are to be exploited. This may happen when developing new interfaces for existing services, if the existing interface cannot be revised.

Many of these considerations overlap with issues discussed in the review in section 2.1. The effects of limitations in the interaction technologies is a very valuable contribution from the interviews, in that it highlights that technology specific details may influence usability and the users' perception of consistency along dimension that are seemingly device independent (e.g. vocabulary and metaphors).

3. A CASE-STUDY OF E-MAIL

3.1 The Aim of the Case-Study

The aim of the case study was to elaborate on a first pass version of the method (which was based on our review of consistency and on the developer interviews), and to derive detailed requirements for further development of the method. An example of a multiple-UI application was documented with respect to the selected consistency dimensions, and we performed an analysis of consistency across the different user interfaces. The documentation of the analysis was used as input for the method development.

We chose e-mail for the case study because there are many e-mail implementations using different types of interaction technologies. E-mail is of interest to us also because it is a service that Telenor develops and delivers to its customers, where several, alternative user interfaces using different interaction technologies already exist.

3.2 The Case-Study

In order to determine which e-mail user interfaces and main user tasks to include, we performed a small in-house survey. As a result of the survey we chose a web user interface, a speech user interface (telephone keyboard input, speech synthesis output) and a text user interface for mobile telephones[1] (SMS messaging) from Telenor's e-mail portfolio. In our survey, 99% of the respondents used Microsoft Outlook™ daily, as it was the standard e-mail application at work for the respondents. We therefore included Microsoft Outlook™ (graphical user interface) for a baseline comparison, even though this is not a Telenor application. Much used e-mail clients, e.g. Microsoft Outlook™, Outlook Express™, Eudora™ and Netscape Messenger™, influence user expectations of how e-mail applications should behave, and as such are important in an analysis of consistency.

To document the different user interfaces we used the main tasks in e-mail usage as a starting point. The main tasks we selected based on the survey were:

- Reading e-mail
- Sending e-mail (new, reply, forward)
- Managing attachments (receive, send)

The user interfaces were documented by describing each main task for each of the e-mail applications (see Table 1) along the dimensions of consistency discussed in section 2.2: sequence, metaphors, vocabulary and brand identity. This was carried out through a simple walkthrough of the user interfaces using the main tasks as scenarios, where the different consistency dimensions were documented for each step in the walkthrough.

[1] For the case study, we evaluated the text user interface using a Nokia™ 6150 mobile telephone.

	Outlook	Web	Speech	Text
Read mail				
Send mail (new, reply, forward)				
Attachments (send, receive)				

Table 1. Documentation for Case-Study Analysis

For each main task, we analysed the different user interfaces with respect to internal and external consistency before documenting and analysing the next task. The problems that were discovered were initially categorised as internal or external consistency problems. However, not all of the problems could be categorised as consistency problems. The problems were equally important design problems, and we documented these problems as well, categorising them as 'other' design problems.

Our case study was *evaluative*, in that we analysed existing user interfaces for consistency with the aim to co-ordinate the interfaces along the selected dimensions of consistency. However, further development of the method should also focus on support for co-ordination in *design*, e.g. when designing alternative user interfaces (in parallel) for a new application, or when designing a new alternative interface for an existing application (that may already have several user interfaces).

3.3 Results from the Case-Study

The analysis in the case study revealed many consistency problems in the user interfaces that we studied, and also helped discover design problems of a more general nature. The case study provided insights as to what the method would require by way of supporting knowledge and techniques in order to be operationalised. Below we have given some examples of the types of consistency problems that were discovered.

A typical problem with respect to the consistency dimension of vocabulary was the use of different terminology for the same object or operation, both within one UI (internal inconsistency) and across the different UIs (external inconsistency).

When we studied consistency with respect to sequence, we focused on the sequence of *logical* operations. The number and sequence of *physical* operations is strongly related to the interaction style, and is difficult to compare across interaction styles.

One simple example of inconsistency with respect to sequence, is when the user wants to reply to e-mail. In Microsoft Outlook™ the “To”-field and “Subject”-field was automatically filled in and the cursor was placed in the text field so that the user could start writing the reply immediately. In the web-based application the “To”-field and the “Subject”-field was also automatically filled in, but the cursor was placed in the “To”-field. The user then had to move the cursor down to the text-field to start writing.

The comparison of UIs with respect to metaphors was more complex. The use (and choice) of metaphor is influenced both by the interaction style and the user equipment. Also, one has to define whether to put more emphasis on consistency between system concepts and real-world concepts or on consistency with respect to one metaphor and how it is implemented in alternative user interfaces. Over time, users become accustomed to the way a particular real-world concept is represented as a system concept. Even though the system concepts may be inconsistent with the real-world concepts, this may be of less significance to users than if the system concepts vary across alternative user interfaces.

In our case study, one example of inconsistency with respect to metaphor was related to how to organise a list of e-mails. Both the graphical user interfaces (Microsoft Outlook™ and the web-interface) uses a vertical list of e-mails. The e-mails are organised as a stack where the most recently received e-mail is placed on top of the stack. The speech user interface, however, organises the e-mails in a horizontal list, which is consistent with a metaphor of a time-scale where time is represented along the x-axis. Users with previous experience of e-mail will normally have an image of a vertical list of e-mails in mind, and the horizontal list may therefore be confusing and cause errors.

In the analysis of brand identity, we found that several names were used for one and the same service. The overall service name was "Epostleser" (e-mail reader), and had two alternative user interfaces. The web-based interface was indeed called "Epostleser", but the speech interface was called both "Epostleser" and "Talende epost" (speaking e-mail) in different parts of the user interface and documentation. This may cause confusion e.g. as to which mail store the users actually operate on. In fact, the two interfaces operate on the same mail store. Also, in the two alternative user interfaces, the company name was presented in very different ways. In

the speech interface, the company name was clearly stated in the welcome message, whereas in the web-based interface it was difficult to tell which company delivered the service.

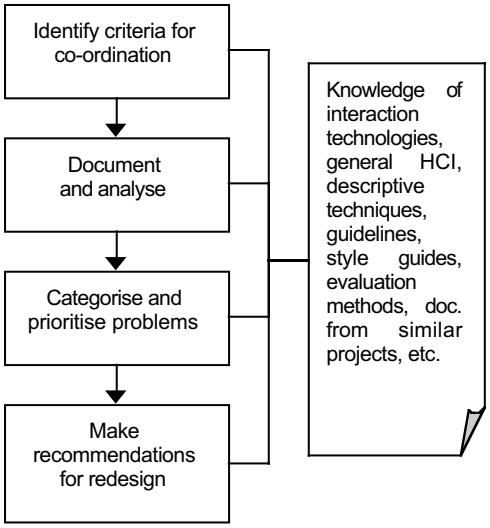
4. mCAT – A PARTIAL SOLUTION

4.1 Status of Current Version of mCAT

The version of mCAT (method for Co-ordination Across Technologies) presented in this section is an attempt at postulating a method for co-ordinating across interaction technologies, using consistency analysis as a tool for the co-ordination. At this stage of development, the method consists of an organisation of developer tasks into four main activities, and the identification of four dimensions of consistency to perform the analysis with respect to. This section also explores further and in more details the requirements for the method. The emphasis has so far been on evaluation, and the use of the method in a design context is only discussed to a limited degree.

4.2 mCAT Overview

Figure 1. The mCAT method



The proposed method is divided into four main activities. Each activity has a goal and requires specific input information. An activity results in one or more products (descriptions, lists of problems, etc.) that may be used in one of the other activities. The method tries to combine new techniques with existing and well-established techniques, and to make them available for the developer. Figure 1 shows a high-level visualisation of the mCAT method.

The method does not require the activities to be carried out in a sequential order, and does not require all the activities to be carried out. However, the model in Figure 1 indicates a preferred sequence of activities as shown by the lines with arrows.

The mCAT method is not meant to be a stand-alone design methodology, and should be configured to work with the development methods that are already being used. The recruitment of existing knowledge and techniques is indicated in Figure 1 by the lines without arrows. A requirement for future versions of mCAT is that it should support the configuration process explicitly.

4.3 Identify Criteria for Co-ordination

In this activity we pose the question: "What are the 'things' that the user interface(s) should be co-ordinated with respect to?" A distinction must be made between an evaluative and design context of using the method.

In an *evaluative* context, existing documentation may be available (e.g. descriptions of existing interfaces for the application, guidelines and style guides) from which criteria for co-ordination can be established. In a *design* context the developer may have to generate (or collect) this documentation to establish the co-ordination criteria. In both cases the developer must identify the alternative user interfaces that should be included in the analysis of consistency, and whether these are complete interfaces or only part interfaces.

The method should support the developer in identifying criteria for co-ordination across the user interface alternatives with respect to the four selected dimensions of consistency: sequence, metaphors, vocabulary and brand identity. At this stage, it is important to recruit knowledge of the target interaction technologies, if possible. The strengths and weaknesses of these technologies may have implications for consistency.

As an example, speech recognisers achieve a much higher recognition rate for command words that sound different (are phonetically distant) than for command words that sound similar (are phonetically close), e.g. 'old/new' would give better accuracy and fewer errors than 'read/unread'. One could hypothesise that even for web-based interfaces words that are distinct (sound different) would potentially work better and lead to fewer errors. Therefore, in this case, to achieve consistency between a speech recognition interface and a web-based interface, the limitations of the speech recogniser should influence the design of the command words (or menu labels), even for the web-based application.

Similar effects can be observed for small-screen interfaces, e.g. small displays that can only display short menu labels. A possible strategy here could be to design commands that have a long form and a short form (e.g. 'New Messages/Old Messages' and 'New/Old'). A certain level of consistency is achieved, while taking into account the display limitations.

However, it is important that the limitation of one of the technologies does not make the other interfaces sub-optimal, as was emphasised in the developer interviews.

4.4 Document and Analyse

In an evaluative context this activity should support the developer in discovering inconsistencies between the different user interfaces. The procedure is equivalent to that described for the case study. The method specifies the use of the main tasks as a starting point for documenting each of the alternative user interfaces (see Table 1 in section 3.2. A consistency analysis is performed with respect to the selected dimensions of consistency.

There are several techniques for identifying the main user tasks, e.g. surveys, usage statistics on existing products, observation of users, market analysis or selection based on experience. The method should support different alternatives.

When documenting the user interface alternatives to be analysed, there are many possible descriptive techniques for the different consistency dimensions. For sequence, some form of procedural description may be appropriate, formal or informal. The vocabulary can in most cases be represented as text, e.g. as a list of definitions, which is what we used for the case study. For documentation of the metaphors that are used in the different alternatives, we did not use a formal approach. For the purpose of the case study, the metaphors were described in prose, as were the elements related to brand identity (e.g. the use of logos, the mention of the company or product name, etc.).

Finding an appropriate way of documenting the user interfaces with respect to the consistency dimensions may be difficult and will depend on the developers' previous experience and knowledge of descriptive techniques. The method should attempt to support the developers in finding appropriate techniques, and give examples and templates for the documentation.

In the case study (evaluative context) the documentation activities and consistency analysis activities were intertwined, even though we attempted to keep the two types of activities separate. This is to a degree consistent with designer studies that have looked at different strategies with respect to breadth-first vs. depth-first approaches, as observed in (Guindon 1990) and also discussed in (Ball and Ormerod 1995).

The breadth-first approach would be to document all main tasks for all interfaces before proceeding to do any analysis. In the case study we ended up documenting one main task for all interfaces, and subsequently analysing the resulting descriptions before proceeding with the next task. This could perhaps be characterised as a mixed approach.

Our experiences from the case study showed that many of the same inconsistencies appeared as more and more main tasks were documented. This may indicate that in a mixed approach it is easier to determine when to stop the process of documenting the interface, thereby saving effort spent documenting the user interfaces. The main requirement is that the method should support both strategies by allowing the developer to move easily between documentation and analysis, and without having to duplicate information.

4.5 Categorise and Prioritise Problems

This activity aims to categorise and prioritise the design problems that have been found. We have defined three categories of problems: internal inconsistency, external inconsistency and 'other' design problems. In the case study, the problems we found could not always be attributed to inconsistency, which is why we included a category of 'other' design problems.

The problems are then prioritised according to relevant criteria, such as how serious the problem is for the user, how much developer effort is involved in solving the problem, how much such an effort will delay the project, etc. In the case study, the prioritisation of problems were not discussed at all, but this is a very important activity that the method should support.

The developer may need to recruit outside knowledge (e.g. on user profiles, general HCI and usability, estimation of developer efforts) to establish criteria for prioritisation. A likely scenario is that the prioritisation activity is a group effort, drawing on potential and representative end users, relevant development teams and managers that are involved in the process.

This approach would seem to be in line with the definition of consistency in (Reisner 1993), where consistency is not only "doing similar things in similar ways", but also "agreement between agents about which

things are similar". Agents in our specific context could be potential and representative end users, developers and managers.

4.6 Make Recommendations for Redesign

In this activity the developer should make recommendations about how to solve the individual, prioritised problems in detail.

On the one hand, inconsistency may be the better solution due to limitations of the interaction technology, or may be deliberate in order to gain the end user's attention. On the other hand, inconsistencies may cause user errors and difficulties in transfer of learning. Knowledge of interaction technologies and general HCI is essential in order to arrive at informed solutions.

In an evaluative context, the output of this activity will be recommendations and possible solutions for redesign to solve the prioritised problems. In a design context, this activity could for example result in a design guide (style guide) or the update of such a design guide. The method should provide support to derive good solutions, e.g. through the availability of design guidelines and knowledge of specific interaction technologies (strengths, weaknesses).

5. DISCUSSION AND FUTURE WORK

The work on mCAT method is still at a very early stage, as discussed in section 4.1. However, the organisation of developer tasks into the main activities of the method has a sound base in findings in the literature review, the expert interviews and in the case study. We have tried to incorporate these findings through deriving and elaborating further requirements for the method.

We have identified some dimensions of consistency that may potentially work across interaction technologies, with some limitations. We limited the number of dimensions of consistency that we addressed in the first version of the method. For future work, the method could be extended to encompass other dimensions that are relevant across interaction technologies.

This paper can not present a complete and operational method for the co-ordination across interaction technologies, but the method may, even in its current rudimentary state, pinpoint some key issues to consider when *evaluating* alternative interfaces for the same application (or service) using different interaction technologies. Future work on the method should attempt to provide support for *design* also, not just evaluation.

One important limitation of our work is that we have made the assumption for our first-pass method that there is only one developer involved in the analysis. We decided to focus on the core issue of consistency for a simplified scenario where one developer analyses alternative user interfaces for a set of functions (e.g. e-mail).

In a real-world situation, this is frequently *not* the case, and the method will in the future have to address the co-ordination of different teams of developers. However, this problem raises a number of organisational and group communication issues that we decided to address at a later stage. Furthermore, a familiar setting of today is the requirement for co-ordinating across several products and development teams using variants of the same interaction technology. Potentially, there are existing strategies and tools that can be modified for our situation (e.g. style guides).

One of the requirements for the method was for it to be effective also for short development cycles, and for it to be used without extensive training. It is a challenge to develop a method within those limitations, and we clearly see the need for a tool to support the method and make it easy to use. However, same as for the method, it is important that the tool is effective and does not attempt to address all problems. The tool should support the application of the method, and not be the method itself.

The different activities in the method need to be described in detail and made operational, e.g. through the development of checklists, template documents and examples, and through the development of the supporting tool. The configuration of mCAT to fit with development methodologies that are already being used must be explicit and operational.

There is a clear need for evaluation of the method: how well does it actually support co-ordination across interaction technologies? We plan to perform an expert evaluation of the method as a start, and after revising the method, to apply it to a real world development effort.

6. REFERENCES

Ball, L. J. and Ormerod, T. C. (1995). Structured and Opportunistic Processing in Design: A Critical Discussion. *International Journal of Human-Computer Studies*, **43**(1), 131-151.

Caulton, D. A. and Dye, K. (1997). Do Users Always Benefit When User Interfaces Are Consistent*?* *In*

Proceedings of the HCI'97 Conference on People and Computers XII , pp. 57-66.

Grudin, J. (1989). The Case Against User Interface Consistency. *Communications of the ACM*, **32**(10), 1164-1173.

Grudin, J. (1992). Consistency, Standards, and Formal Approaches to Interface Development and Evaluation: A Note on Wiecha, Bennett, Boies, Gould, and Greene. *ACM Transactions on Information Systems*, **10**(1), 103-111.

Guindon, R. (1990). Designing the Design Process: Exploiting Opportunistic Thoughts. *Human-Computer Interaction*, 5305-344.

Kellogg, W. A. (1989). The Dimensions of Consistency. *In Coordinating User Interfaces for Consistency (Ed, Nielsen, J.)*. Academic Press, New York, pp. 9-20.

Löwgren, J. and Lauren, U. (1993). Supporting the Use of Guidelines and Style Guides in Professional User Interface Design. *Interacting with Computers*, **5**(4), 385-396.

Nielsen, J. (1986). A Virtual Protocol Model for Computer-Human Interaction. *International Journal of Man-Machine Studies*, **24**(3), 301-312.

Nielsen, J. (Ed.) (1989a) *Coordinating User Interfaces for Consistency*. Academic Press, London.

Nielsen, J. (1989b). Executive Summary: Coordinating User Interfaces for Consistency. *In Coordinating User Interfaces for Consistency (Ed, Nielsen, J.)*. Academic Press, New York, pp. 9-20.

Reisner, P. (1981). Formal Grammar and Human Factors Design of an Interactive Graphics System. *IEEE Transactions on Software Engineering*, **7**(2), 229-240.

Reisner, P. (1993). APT: A Description of User Interface Inconsistency. *International Journal of Man-Machine Studies*, **39**(2), 215-236.

Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing Company, Reading, Massachussets.

Vanderdonckt, J. M. and Bodart, F. (1993). Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. *In Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems* , pp. 424-429.

Wiecha, C. (1992). ITS and User Interface Consistency: A Response to Grudin. *ACM Transactions on Information Systems,* **10**(1), 112-114.

Wiecha, C., Bennett, W., Boies, S., Gould, J. and Greene, S. (1990). ITS: A Tool for Rapidly Developing Interactive Applications. *ACM Transactions on Information Systems*, **8**(3), 204-236.

