A Structured Contextual Approach to Design for All

Chris Stary

Communications Engineering, Department of Business Information Systems
University of Linz, Freistädterstraße 315, A-4040 Linz, Austria

stary@ce.uni-linz.ac.at

Abstract. Although a variety of concepts have been published to implement the design principles for user interfaces for all, there is still a lack of techniques applicable for structured development of this type of user interfaces. This paper deals with an approach that has been developed in the course of an industrial design project. It suggests to gather design options and structure the design process through a formal decision making procedure, hence increasing the maintainability of deign solutions and products this way.

1. INTRODUCTION

User Interfaces for All is a concept that targets towards universal accessibility of information. Regardless of their role, skills, requirements, experiences, and abilities humans should be able to interact with information systems in an accurate way (Stephanidis et al., 1998). User interfaces for all focus on the *pro-active* consideration and incorporation of the diverse requirements throughout the development life-cycle rather than on the development of specific solutions for the provision of accessibility to specific user categories (*reactive* approach). Pro-activity addresses the accessibility of user interfaces at design time, in order to finally guarantee the utmost utilisation of an artefact. Hence, pro-activity also requires implementation-independent descriptions (specifications) of any system, in order to check whether the intended users (or user group) could be empowered through the artefact or not. Consequently, design support has to provide not only the representation of user characteristics but also the assignment of these characteristics to particular styles of interaction or metaphors (requiring interaction styles) (Stary, 1997).

In addition, artefacts should not exclusively be designed for the sake of implementing hard- and software, thus focussing on an engineering perspective ('to put things to practice'). They should also be designed in the sense of conceiving and planning a socio-technical system (as suggested in Beyer et al., 1998, p.3). As such, designers have to understand more than a variety of requirements, guidelines, and representation techniques, namely, how to handle different perspectives and sources of knowledge, and to develop an open design representation/model consistent with users' goals and interaction contexts. Unfortunately, the dynamics involved in integrating several factors into a usable product elude designers. Yet designers need to understand these dynamics in order to create software that is truly useful to and valued by users (Newman et al., 1995). In practice, this means, for instance, that information types (i.e. codality of information) do not only have to be considered at the syntactical design layer, but also at the semantic and pragmatic layer, since the encoding of information plays a crucial role in understanding content and behaviour of artefacts.

The process of mapping complex, contingent human behaviours of information processing to rule-bound events and properties of accurate interfaces is an extremely challenging task. Challenges occur in such core design areas as representation, methodology, shared language and communication. In this paper key pulse points among these challenges are addressed: the development of design spaces and facilitating design decisions. A procedure for unifying isolated views about interaction elements and styles as well as design objectives and options is introduced. It does not only facilitate communication and decision making in product development projects but also the traceability of the design process itself.

We first review the design 'ingredients' and their embodiment into the design process when developing user interfaces for all (section 2). We then introduce the procedure we followed to construct design solutions (section 3). The paper concludes with reviewing some results and an outlook for further research activities (section 4).

2. DESIGN, KNOWLEDGE, AND DECISION MAKING

In this section we do not only review existing work in the field of complex product development involving user interface design for all, but also refine some of the requirements addressed above for a structured approach to design for all.

2.1. User Characteristics

Taking into account user characteristics design has to be understood as a process that views knowledge about users and their involvement in the design process as a central concern' (Preece, 1994). It does not only require communication between end users and designers, but also a common understanding among developers. Assume the design of a publicly available information kiosk at a railway station. In case the terminal and the software should be designed user-centred, a variety of modalities, objectives to use that information, and alternatives have to be discussed. Although it is agreed that today's common practice should focus on user needs, there is neither consensus about

(i) how to involve end users, nor about
(ii) how the migration of user involvement into software engineering activities has to occur.

'Much of the existing advice is complementary, not contradictory, but little attempt to integrate separate facets into a coherent methodology has been made' (Gardner, 1991). This statement points to methodological problems, although the economic proof of user-oriented development has already been made by that time by Mantei et al. (1988). One reason for these problems might be that user-centred design has to be 'done in concert with engineering realities of functions to be provided, schedules to be met, and development costs to be managed.' (Karat et al., 1991) Surveys of design and development practice do not provide sufficiently insights, since they have been focused mostly on the underlying principles for application development rather than on the actual process of application design.

2.2. Technology

In case, different technologies, e.g. telecommunication and information technology, have to be migrated in the course of product development, one of the underlying principles that requires the communication of design knowledge is the demand for co-operation among developers from different disciplines. Here, the same rules as between designers and users have to be applied. Different 'kinds of instruments', 'different kinds of objects', and different aims of work (Bodker et al., 1991) might have to be discussed and mutually tuned. Experts are experienced in their domain, but need to be co-ordinated in collaborative design efforts (Erickson, 2000). For instances, Bodker et al. (1991) have found out that software development experts have to suspend their expert status in the dialogue with other experts, since different cultures of work have to collaborate throughout design. Both have to develop a common language. That language can neither be the language of experts, since it is too specialised, nor everyday language, since it remains too ambiguous with respect to semantics (Vollmerg et al., 1992).

Typically in product development, designers are guided by restrictive *principles* and paradigms. For instance, Grudin (1991) has identified some these principles for user interface design - underscoring that purported support of users has not been proven. These principles are design simplicity, consistency with a real-word analogue, and anticipation of low-frequency events. In addition a product-oriented perspective still prevails in industry, viewing software as a stand-alone product in contrast to a process-driven approach, as e.g., defined by Floyd (1987). As a stand-alone solution, it abstracts from the underlying system characteristics and assumes a predefined, in most of the cases, idealised context of use, thereby allowing requirements to be specified before they are implemented. Hence, the conventional development paradigm leads to a system that is designed by (several different) specialists in accordance with technical and economic criteria set by management, but with little reference to current and future users and contexts of use. This design is then implemented - with limited scope for modification, but, e.g., as advocated by the early schools of software engineering (Jackson, 1983), reducing labour processes to technology-driven information processes (Diaper et al., 1992). Finally, in particular software designers are interested in immediate effects within their structural concepts (Ropohl, 1979), i.e. what can be applied to construct an artefact that works (engineering perspective), and final assessment in the context of use (Flores et al., 1988).

2.3. Tasks

However, information systems accessible for all should not be based on idealised processes that are performed with the help of or by artefacts (Bodker, 1998). Nor should design representations be considered to be mappings of current or envisioned (work) situations and/or applications. Rather they should serve as containers for ideas, carrying their own context, and evolve iteratively with continuous improvements (Floyd, 1987). They should cover the entire design space, capturing design options and supporting structured decision making. This conception has been underlined through recent findings in the field of requirements tracing (Jarke, 1998). It has been recognised that user needs are changing permanently, but the need for consistent system development and evolution remains. Design is considered to be crucial for requirements specification.

2.4. Traceability and the Design Process

Traceability of development has been emphasised, but rarely addressed from a methodological perspective in the user interface community. One reason might lie in the fact that user interface builder encapsulate the behaviour of dialog elements and styles. As such, task-related behaviour is mapped on to predefined sequences of states of dialog elements (being part of the platform) without further specification of their behaviour. However, traceability enables repeatability of software development processes - a stage addressed by level 2 of the software Capability Maturity Model (Humphrey, 1990). At higher levels, comparing traces to process plans is required - a feature that is also based on transparent and traceable processes. Data from traceability analyses provide evidence that poorly developed development organisations (also termed low-level users) are not very likely capable to meet all customer requirements and to produce systems that are easy to maintain, whereas high-level users let customers and end users participate, and capture traces across products and process dimensions (Ramesh, 1998).

2.5. Complex Requirements and Design as a Process of Transformation of Knowledge

Collection and management of complex requirement data without loosing detail have been addressed by some development methods, such as contextual design (Beyer et al., 1998). 'Contextual design is an approach to defining software and hardware systems that collects multiple customer-centred techniques into an integrated design' process.' (ibid., p.3) Unfortunately, it exclusively makes data gathering from potential users the base criteria for deciding on how the system's structure and behaviour should look like. This strategy can only be implemented in case users are able to envision their access to information in some predefined way, e.g. performing particular tasks. But, what if the task domain cannot be structured well or the vast majority of users are not known in advance, e.g., in case of developing a novel series of products? These cases can only be handled through flexible design spaces, flexible architectures, and structured procedures to come up with those.

When design is understood as a continuous process of knowledge transformation, a step-by-step procedure allows to move towards a solution. To that respect, Ludolph (1998) suggests to design by successively transforming task/object models in the course of developing context-sensitive user interfaces. The addressed process is based on:

- background knowledge, such as requirements and real-life scenarios,
- an essential model, which is a high-level description of the application's fundamental functions without reference to technology or how the user will actually perform them,
- a user's model, i.e. the concepts, objects, and tasks as seen from the user's perspective, free of presentation and interaction elements, and finally,
- a completed design, this is how a person will see, think about, and interact with the application, but including the elements for interaction.

As can be seen, the context is kept until the last step, namely, the design of the artefact. Following this procedure ensures a user perspective on the flow of control at the specification level. If implemented this way, it will be perceived correspondingly at the user interface.

2.6. Diversity of Interaction Styles

Another issue which has to be discussed in the context of this work is *multi-modality*. User interfaces for all do not only have to provide a variety of ways to interact with information systems, but also features to switch between these modalities, e.g., between visual and acoustic output. As a consequence, design spaces have also to capture multiple styles of interaction, either for in- or output. Different styles of interaction might have to be combined in a variety of ways. Traditionally, multi-modal systems process combined natural input modes (speech, pen, touch, manual gestures, gaze, and head and body movements) in a co-ordinated manner, preferably with multimedia system output. This type of interfaces represents a new direction for development. It also requires a research-level paradigm shift away from conventional WIMP interfaces towards providing users with greater expressive power, naturalness, flexibility and portability.

Focus of multi-modality research has been the technical integration of signals of different sources, e.g., Cohen et al. (1997), rather than conceptual or methodological issues for development. For designing user interfaces for all, a wider understanding of multi-modality is required. In general, it represents the use of different senses and channels of communication (auditory, tactile etc.). It is strongly related to *multi-codality* which addresses the issue of how to use different codes or symbol systems to encode and present information (textual, graphical, pictorial etc.). Multi-modality tries to map elements and styles from human face-to-face communication to in- and output features at the user interface. From the input side, in particular, voice, gestures, and facial expressions are of interest. From the output side, anthropomorphic functions, avatars, animated agents, speech, and virtual assistants are elementary features besides the traditional ones, such as windows, icons asf. A more conceptual understanding of the capabilities and the interplay of multi-modal dialog elements should enhance the design space.

2.7. Decision Making

Enhancing the design space impacts decision making. The larger the set of alternatives to provide solutions, the larger the need for structured and transparent decision making. As we know from the history of software engineering, decisions that are not made in the course of design and detailed specification, are made through programmers when coding. Hence, a procedure for a structured design-for-all-process has to also to support decision making. With respect to designing interactive systems, only few techniques have been applied successfully. The Questions, Options, and Criteria (QOC)-notation (McLean et al., 1991) and a corresponding procedure help to formalise and record decision making. It forces developers 'to standardise and document design issues (questions) in deciding which alternatives (options) to keep' (Simpson, 1998, p. 257). The procedure also helps in structuring relationships between options and their context of use, namely through making explicit the criteria for evaluating the options. As such, QOC turns out to be an ideal candidate for long term product development. Simpson concludes, 'a formal decision-making method - most likely recorder after than during a design session (so as not to stifle creativity) - would help with maintainability of the interface design over time.' (ibid.)

3. TOWARDS AN EMBEDDED DESIGN SPACE ANALYSIS

After having introduced the major ingredients for a structured and open approach to the design of user interfaces for all, in this section we introduce first steps towards the definition of the Embedded Design-Space-Analysis (E-DSA) procedure. We give the concept and detail its use from experiences in an industrial design project.

The sample case concerns the extension of a set-top box, as e.g., conventionally used for TV appliances, with communication facilities, in order to have a personal communicator for home and mobile use. The envisioned scenario of use comprises several facilities:

- Internet-connection to a provider via the set-top box
- Digital fax, phone calls, and emails as inputs via Internet or phone
- Mobile-phone screen or TV screen for output
- Remote control from TV or the mobile-phone keypad for control
- Keyboard for data input.

E-DSA targets towards the structured handling of design spaces with respect to interaction modalities and decision making when selecting design options. It comprises 3 steps: (1) *Set Up of Interaction Space*. In this step the available elements and styles for interaction, including the type of information that can be processed (codality of information) are captured. This knowledge can be evaluated according to different perspectives, and assigned to metaphors for designing intuitive features for interaction. (2) *Set Up of Task Space*. The set up of the task space captures declarative (the 'what') as well as procedural knowledge (the 'how'). Objectives are restated in terms of tasks. The context of task accomplishment is detailed in terms of objects, operations on those, and constraints concerning tasks and their accomplishment. (3) *Contextual Exploration and Analysis:* The specification of design solutions is performed through assigning dialog elements and styles to task procedures. It is based on structured decision making, namely, selecting options based on design criteria, stemming either from usability engineering or the constraints given for task accomplishment.

Step 1. Interaction Space Set Up. For the set up of the interaction space the framework proposed in Stary (1996, p. 129, p. 179) has been extended with state-of-the-art styles of interaction, since other frameworks either lack the required level of granularity, e.g., such as the one proposed in Newman et al. (1995, p. 294ff), or do not take into account the characteristics of use, such as the channels of communication for interaction. Finally, metaphors and characteristics of modalities above technology are encountered rarely through existing frameworks. However, both are of crucial importance for designing user interfaces for all. Adequate metaphors facilitate handling interaction devices, and generic characteristics allow an implementation-independent view on the development knowledge. In the following we detail this set-up process. It comprises the two sub steps described subsequently.

Step 1a. Contextual Modality Specification. According to several perspectives the modalities of interest have to be captured. Initially, the elementary (key-modal) styles of the design space are specified. The *technical* perspective is addressed through generic structure and behaviour elements as well as categories of use (control, navigation, data in/output), such as shown in Lee (1983) for GUIs. This type of descriptions turned out to be extremely useful when designing compatible products, e.g., as recently shown in the field of browser development, however, at the syntax layer (http://power.eng.mcmaster.ca/alden/ti.htm).
Table 1 contains menu and window descriptions at the generic layer. Those descriptions have to be developed for designers, not for programmers. As such, they are abstractions that hold

across platforms and various implementations. They might become resident parts of a design space. They are easy to (re)use and to handle for further developments.

Modality	STRUCTURE	BEHAVIOUR	CONTEXT OF USE
Menu	Title Bar Option Field	Open Close Highlight	Control Navigation
Window	Title Bar Scroll Bar Work Area Control Area Tool Bar	Open Close Quit Resize Back/Foreground	Data in/output

Table 1. Examples for generic interaction style descriptions

The *human-oriented perspective* as well as the *application-oriented one* are addressed in the following. Table 2 and 3 (upper-bound entries) contain an elementary style of interaction (menus) and a composed one (GUIs) that are specified in terms of contextual items. Table 2 shows the involved channels for interaction and the required user actions as well as the provided feedback to inputs by an interactive computer system. A menu might be perceived visually on the screen and manipulated through manual selection, directly visible on the screen. Table 3 shows details with respect to input-output behaviour, capabilities for information codality, required devices for interaction, and guidelines. A menu can be used as control input device to navigate through a task hierarchy. It might also be used as a data input device, in case its entries correspond to a set of valid data items. There exists graphical, text-, and audio-based menu types. Devices for menu interaction range from touch screens to a micro and speakers. The entries for GUIs will be addressed in step 1b, since in step 1a only elementary styles, such as interaction via menus, icons, windows, command languages are captured.

Type of relation-ship to user Modality	PERCEPTION	HANDLING	FEEDBACK TO INPUT
Menu	Seeing Hearing	Selection (Visual) Voice (Acoustical)	Visual Acoustical
Graphical User Interface (GUI)	Seeing Hearing	Window Management	Visual Acoustical

Table 2. Examples for specifying key-modal (step 1a) and composed interaction styles (step 1b) with respect to involved user / system actions

Step 1b. Cross-modality Specifications. This steps targets towards an accurate description of those combinations of modalities that should be considered for the design process. Firstly, composed interaction styles are captured in a cross-modality matrix (see also table 4). Secondly, the contextual information has to be acquired analogously to each of the elementary styles (see also table 2 and 3, lower-bound entry).

Further Parameter of Use Modality	TYPE OF INPUT	CODALITY	REQUIRED DEVICE	GUIDELINE / CONSTRAINT
Menu	Control Data only as a list	Text Graphics Audio	Screen (incl. Touch) Keyboard Pointing Device Speakers Speech Recogniser	List of options <9 for symbol/text entries
Graphical User Interface (GUI)	Control Data (Window)	Text Graphics Audio	Visual Display Unit (VDU) Pointing Device	Provide options for tiled/overlap-ping windowing

Table 3. Examples for detailing key-modal (step 1a) and composed interaction styles (step 1b) with respect to their application

Modality	Menu	Window	Icon
Menu	Menu style GUI, in combination with Windows, Icons, Pointing Devices	Plain screen GUI, in combination with Icons, Menus, Pointing Devices	Graphical menu Symbolic interaction GUI, in combination with Windows, Menus, Pointing Devices
Graphical User Interface (GUI)	Enabler (Control)	Enabler (Data)	Enabler (Control)

Table 4. Part of the cross-modality matrix (initial activity of step 1b)

Step 1c. Concept/Metaphor Assignment. The final activity in step 1 is the assignment of metaphors or interaction concepts (paradigms) to the specified styles in step 1a and 1b. Usually, this assignment is performed at the level of interaction styles involving more than one modality, as table 5 shows. For our sample case, table 6 and 7 comprise the menu specification. As can be seen, several types of menus (textual, acoustical, and graphical) are part of the design space for the set-top communicator. It also becomes evident from the list of constraints in table 7 that the product setting requires specific restrictions to the use of menus. In case of using a mobile-phone display as an output facility, due to space limits, a menu in list form (e.g., pop-up) must not contain more than 3 entries. When there are more than 3 options to be displayed, another form of presenting control data to users has to be used.

As a result from step 1 a variety of constellations given through the interaction design space becomes available. In E-DSA, so-called *descriptors* have been developed to allow an integrated perspective on an interaction style. For instance, the descriptor P/seeing-H/selection-F/VDU-unit capture all human-oriented elements for visual interaction via menus, with P: Perception, H: Handling, F: Feedback. Descriptors turned out to be useful to describe all possible constellations within styles (technological perspective). They do not only enable an integrated view, but also take into account variations within styles, such as the coupling of acoustical presentation of menu options with visual selection of options. Finally, interaction can be described through descriptors for codalities and metaphors.

Interaction Concept/ Metaphor Multi-Modality-Constellation	DIRECT MANIPULATION	HANDY	PORTAL
GUI	Enabler	Partial Enabler	Enabler
Virtual Reality	Enabler	Enabler	Partial Enabler

Table 5. Part of the concept/metaphor – modality-matrix (step 1c)

Type of relationship to user Modality	PERCEPTION	HANDLING	FEEDBACK TO INPUT
Menu	Seeing Hearing	Selection (Visual) Voice (Acoustical)	Visual Acoustical

Table 6. Step-1a results with respect to menu interaction in the set-top-communicator case

Further Parameter of Use Modality	TYPE OF INPUT	CODALITY	REQUIRED DEVICE	GUIDELINE / CONSTRAINT
Menu	Control Data only as a list	Text Graphics Audio	Screen (also Touch) Keyboard Pointing Device Speakers Speech Recogniser	List of options <9 for graphical/text entries IF #Options > 3 AND output = handy display options must not displayed in a list

Table 7. Step-1a results with respect to menu interaction in the set-top-communicator case

Step 2. Task Space Set Up. The set up of the task space targets towards the specification of the essential model. According to Constantine (1995), the essential model is to define the tasks users might perform without describing how each of the tasks is actually performed. It rather describes user's intentions. The model consists of the

- tasks a user wants to accomplish,
- involved objects and operations that comprise those tasks
- relationships among those objects
- one or more use cases for each task.

The tasks should be named, include information on required in/outputs, volumes, frequency of execution, functional roles that performs them, and all known constraints.

This model can either be generated from scratch, e.g., in case of a new product or extracted from background information, such as documents describing organisational details. The core activity at that stage of design is the embodiment of real-life scenarios in terms of tasks. Part

of each scenario is a set of objectives. The objectives state what the scenario is trying to accomplish. They also might refer to objects and information representing the context of task processing. The objectives are then restated as tasks involving one or more objects/data. Information in the scenario about the tasks as stated above (in/outputs, constraints etc.) are listed with the related tasks. From the use case description we already derive procedural information for the E-DSA-task space.

The specification of the essential model follows a certain procedure: (1) Objective(s) identification; (2) Restatement of objectives in terms of tasks; (3) Context specification of tasks; (4) Path definition(s) for accomplishment, (5) Object definitions; (6) Operation definitions in accordance to objects and paths. In our sample case, the objectives have been captured at a macro- and a micro-layer. The macro layer comprises global goals, such as to enable a single point of communication in a household with telecommuting facilities. At the micro-level objectives have been addressed that can be easily mapped to tasks. Below a sample scenario (in the sense of Carroll, 1995) is given that enables the identification of tasks as well as use cases, as required for essential model construction:

A sales person checks her mail after coming home from a business meeting. With respect to her job, she has to book a flight from Vienna to Munich for the next day, and to confirm the meeting on the next day to her manager and the partners of the meeting in Munich. Booking the flight is done over Internet through information agents, and after the flight confirmation and the transmission of details (ground transportation, check-in etc.) the meeting can be confirmed via email.

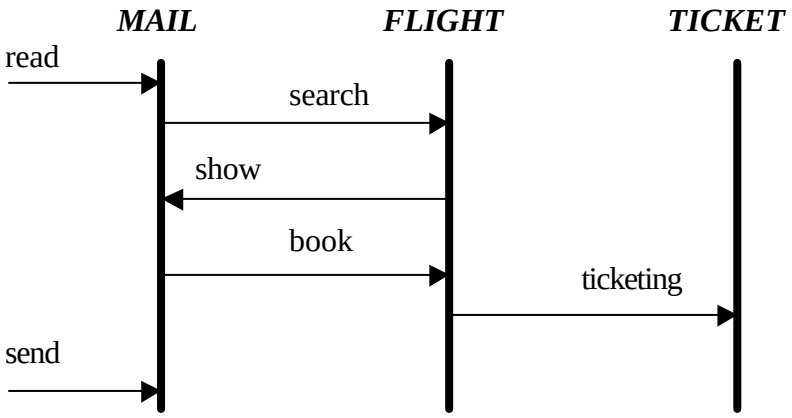

Figure 1. Object-specific workflow of sample scenario

The tasks involved are: checking mail, booking flight, confirm meeting. The context is given through the job description and the telecommuting environment the user is part of. The causal and temporal relationships between the activities determine the path(s) to be followed for successful task accomplishment. The objects involved in task accomplishment (see figure 1) are highly interrelated in that case, since the meeting data are part of the flight data, and the user data are common to the booking and mail task. They identified objects are: mail, flight, ticket. Its operations are derived from the set of options available by mail systems (read, send, attach, etc.) and booking systems (search for flight, select flight offer, book, ticketing etc.).

Step 3. Contextual Exploration and Analysis. The assignment of dialog elements and styles to task procedures is based on structured decision making, i.e. through selecting

options based on design criteria. We use the experiences from applying QOC (Questions, Options, and Criteria) to handle the design space. According to MacLean et al. (1991) Questions identify key design issues. Options provide possible answers to the Questions. Criteria enable the assessment and comparison of Options. For design space analysis (which is understood as structured decision making in the course of specifying a technical artefact) the most important elements are the criteria. They stand for the desirable properties of the artefact and requirements that must be met. As such, they clarify the objectives of the design (process) and establish a ground against which the Options are evaluated.

In E-DSA we distinguish Fundamental Questions and Specific Questions, in order to distinguish between the context and the core of an artefact. *Fundamental Questions* are considered to address design issues that have to be handled regardless of the case at hand. In the following a list of selected fundamental design questions (F-Questions) is given:

- Are there metaphors available that can be applied for control and task accomplishment according to the scenarios at hand? (F-Qu1)
- Which features enable modality and/or codality switching? (F-Qu2)
- Which scenarios might lead to / require switching between modalities? (F-Qu3)
- How can computer-(il)literate users being supported? (F-Qu4)

As can be seen from this short list, this type of questions addresses the most essential features an user interface for all should have. F-Qu1 addresses learnability, ease of use and user conformity. A typical metaphor for the sample case is the mail metaphor for US-users, displaying the letter box according to the state of incoming or outgoing mails. F-Qu2 and F-Qu3 encounter for users with different abilities and needs through asking for the provision of different forms of information presentation and interaction. This way, the adaptability of the artefact is brought into play. F-Qu4 addresses all the previously mentioned principles of usability engineering, since it focuses on the support of novice and experienced users. Both types have to expected for user interfaces utilised by all.

Specific Questions (S-Questions) deal with modalities, functional features and their intertwining. With respect to *functional features* the configuration management for different versions of a product might look like shown in figure 2 for the case at hand. This QOC-application shows the exploration of the design space with respect to a version of the set-top communicator that does not offer fax communication, thus, restricting the access to mail and phone facilities (for the sake of easy-to-learn product features). Once a particular design option is discussed, follow-up questions in the device context have to be discussed, such as shown for option O2 (given in figure 2) in figure 3. According to the criteria, again the variety of features is concerned, and, however, this time the speed is relevant, since attachments of mails might effect the efficiency of communication. With respect to *presentation* figure 4 shows a typical constellation of QOC, namely for the scenario after reading the mail and looking for a proper flight.

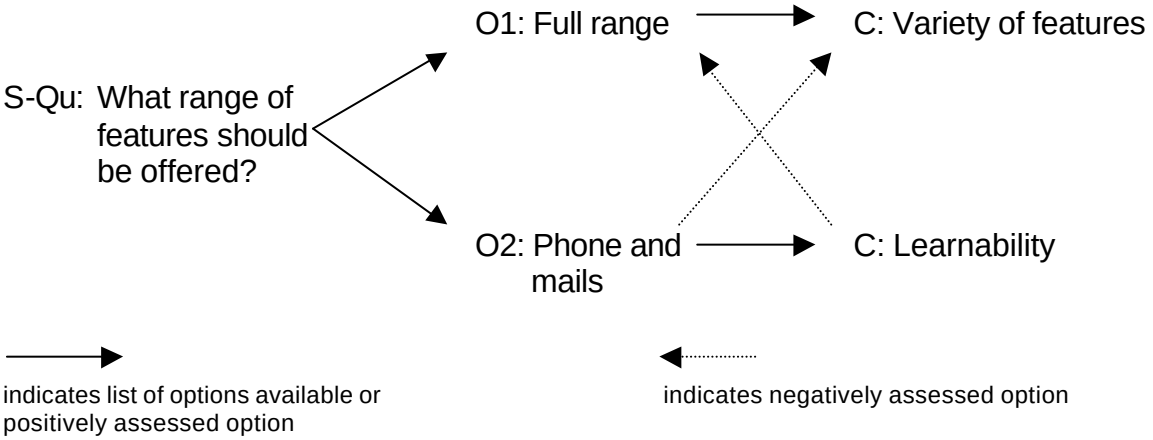

Figure 2. Sample Specific Question with respect to functional features

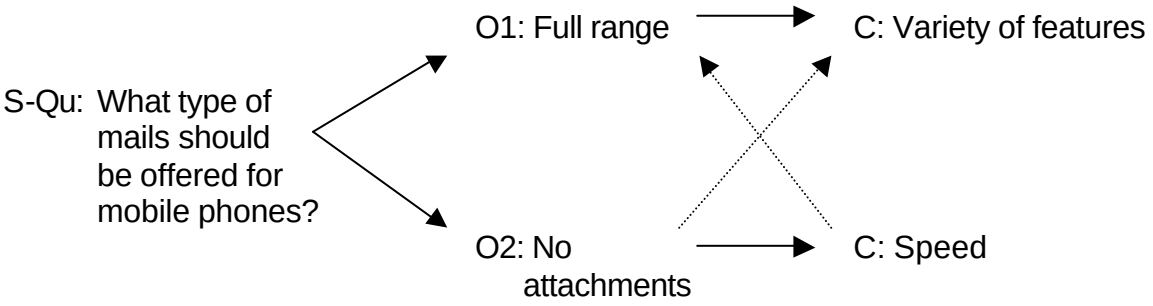

Figure 3. Sample Follow-Up Question with respect to functional features

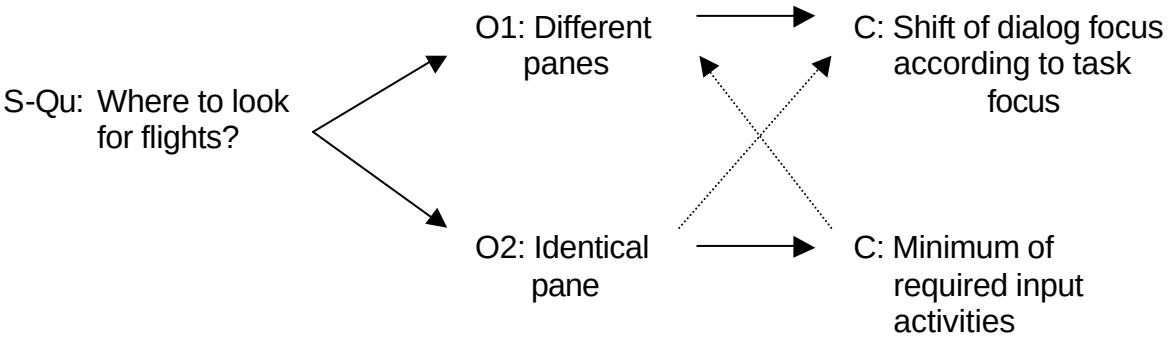

Figure 4. Sample Specific Question with respect to interaction features

It has to be decided whether control inputs should be minimised (as e.g., required to achieve task conformance) at that stage of task accomplishment. Since this step shifts the focus of task accomplishment to flight booking (after reading mails) the user might get lost, e.g., since he/she does not find the way back to the mail tool after having booked a flight. In order to resolve this issue, a contextual constraint might be applied, namely, to use an identical pane for flight booking and e-mailing, only when the user is experienced in handling several, probably different, tools at the user interface at the same time. Using QOC, a Support Argument can be assigned to option O2, when literate users should find dedicated support to that respect. It can also be assigned as a Challenging Argument to the same option, since illiterate users might experience troubles when identical panes are used for different tasks.

Since QOC and Design Space Analysis per se do not replace a structured representation of results, but rather support the process of design, namely how to achieve context-sensitive results, in E-DSA this issue has to be tackled. Presentation is based on the results of step 1 and 2. Thus, the results of decision making are represented in the context of interaction styles. Table 8 shows part of the structured assignment of tasks and objects to modalities. Finer granularity can be achieved through the use of descriptors, showing which interaction features for control, navigation, and/or data input correspond to which activity for accomplishment.

In case execution paths cannot be mapped directly onto states of dialog elements, as e.g., in case of lacking interaction platforms, additional specifications, e.g., state-transition diagrams have to be developed, in order to specify the dialog sequence for task accomplishment.

Design Elements **Modality**	**TASK**	**DATA**	**DEVICE**
Menu	mail	---	Handy: round-about TV-screen: list
Graphical User Interface (GUI)	mail booking	mail ticket	TV-screen: window

Table 8. Task and data assignment to interaction styles for the set-top communicator

3. CONCLUSIONS

For designing user interfaces for all, both, complex product features as well as following the strategy to provide a consistent line of development over product generations, require maintenance techniques at the specification/representation level. In addition, the design process has to be supported in a way, that structured decision making is enabled. As such contextual design spaces provide the means for capturing design-relevant knowledge, and arrange it in an integrated, but still flexible way. Although existing design techniques emphasise particular aspects, such as representing tasks, the entire set of activities for product design has not supported in a consistent way before. We suggest to start out with information gathering and structuring, and come up with contextual specifications.

Through proper representation and decision making techniques, as Bannon (1997) demands, the generative uniqueness of specific disciplinary perspectives can be kept while still coming to agreement about a common object of design. However, before attaining Bannon's ideal of unity in difference, developers must understand the source of their differences -- tracing their divergent views to the fundamental issues of representation, methodology, and an insufficiently shared language. The introduced steps towards a design space analysis embedded in the artefact's context enable to trace the design solution(s) back to the requirements (objectives and tasks), and the facilities for interaction.
E-DSA (Embedded Design Space Analysis) comprises 3 steps. Step 1 leads to a set up of the interaction space, i.e. the modalities available for user interface design. Not only the available elements and styles for interactions, but also the type of information that can be processed (codality of information) are captured. This knowledge is refined according to different

perspectives (humans, technology, and organisation), and finally assigned to metaphors for designing intuitive features for interaction. In step 2 the task space to be supported is specified. Objectives are restated in terms of tasks. The context of task accomplishment is detailed in terms of objects, operations on those, and constraints. In step 3, the spaces set up in step 1 and 2, are explored and evaluated against design criteria. First, fundamental issues for interfaces for all, such as the capability to switch between modalities, are checked. Secondly, modality- and task-specific issues are analysed. Finally, dialog elements and styles for interaction are assigned to tasks and the procedures for accomplishment. In E-DSA the QOC (Question, Options, and Criteria) notation is used for contextual specification and for documenting the decision making procedure. The applied design criteria either stem from usability engineering or the constraints given for task accomplishment.

E-DSA has been applied successfully in the course of industrial design projects. Its further development will focus either on the steps and their mutual tuning, in order to come up with proper software support, in particular for convenient manipulation of design knowledge according to the E-DSA procedure.

References

Bannon, L.: Dwelling in the „Great Divide“, in: Social Science, Technical Systems, and Cooperative Work: Beyond the Great Divide, eds: Bowker, G.; Star, S.L; Turned, W; Gasser, L., Lawrence Erlbaum Associates, Mahway, NJ, pp. 355-378, 1997.

Beyer, H.; Holtzblatt, K.: Contextual Design. Defining Customer-Centered Systems, Morgan Kaufman, San Francisco, 1998.

Bodker, S.; Gronbaek, K.: Cooperative Prototyping: Users and Designers in Mutual Activity, in: Journal of Man-Machine Studies, Vol. 34, pp. 433-478, 1991.

Bodker, S.: Understanding Representation in Design, in: Human-Computer Interaction, Vol. 13, No. 2, pp. 107-125, 1998.

Carroll, J.M. (ed.): Scenario-based Design. Envisioning Work and Technology in System Development, John Wiley, New York, 1995.

Cohen, P.; Johnston, M.; McGee, D.; Oviatt, D.; Pittman, J.A.; Smith, I.; Chen, L.; Clow, J.: Quickset: Multimodal Interaction for Distributed Applications, in: Proceedings 5th Int. Multimedia Conference, ACM, pp. 31-40, 1997.

Constantine, L.L.: Essential Models, in: interactions, Vol. 2, No. 2, pp. 34-46, April 1995.

Diaper, I. Addison, M.: Task Analysis and Systems Analysis for Software Development, in: Interacting With Computers, Vol. 4, No. 1, pp.124-139, 1992.

Erickson, Th.: Lingua Franca for Design: Sacred Places and Pattern Languages, in: Proceedings DIS’2000, ACM, 2000.

Floyd, Ch.: Outline of a Paradigm Change in Software Engineering, in: Computers and Democracy, eds: Bjerknes, G., Ehn, P.; Kyng, M., Avebuy, Aldershot, pp. 191-211, 1987.

Flores, F.; Graves, M.: Computer Systems and the Design of Organizational Interaction, in: ACM Transactions on Office Information Systems, Vol. 6, No. 2, pp. 504-513, 1988.

Gardner, A.: An Approach to Formalized Procedures for User-Centered System Design, in: Human Factors for Informatics Usability, eds: Shackel, B.; Richardson, S., Cambridge University Press, Cambridge, 1991.

Grudin, J.: Systematic Sources of Suboptimal Interface Design in Large Product Development Organizations, in: Human-Computer Interaction, Vol. 6, pp. 147-196, 1991.

Humphrey, W.: Managing the Software Process, Addison-Wesley, Reading, MA, 1990.

Jackson M., System Development, Prentice Hall, New York, 1983.

Jarke, M.: Requirements Tracing, in: Communications of the ACM, Vol. 41, No. 12, pp. 32-36, 1998.

Karat, J.; Bennet, J.L.: Using Scenarios in Design Meetings, in: Taking Software Design Seriously, ed.: Karat, J., Academic Press, London, 1991.

Lee, G.: Object-Oriented GUI-Application Development, Prentice Hall, Englewood Cliffs, 1983.

Ludolph, M.: Model-based User Interface Design: Successive Transformations of a Task/Object Model, in: User Interface Design: Bridging the Gap from User Requirements to Design, CRC Press, Boca Raton, FL, ed.: Wood, L.E., pp. 81-108, 1998.

Mantei, M.; Teorey, T.: Cost/Benefit Analysis for Incorporating Human Factors in the Software Lifecycle, in: Communications of the ACM, Vol. 31, No. 4, pp. 428-439, 1988.

MacLean, A.; Young, R.; Bellotti, V.; Moran, T.: Questions, Options, and Criteria: Elements of Design Space Analysis, in: Human-Computer Interaction, Vol. 6, pp. 201-250, 1991.

Newman, W.M.; Lamming, M.G.: Interactive System Design, Addison Wesley, Wokingham, 1995.

Preece, J.: Human-Computer Interaction, Addison-Wesley, New York, 1994.

Ramesh, B.: Factors Influencing Requirements Traceability Practice, in: Communications of the ACM, Vol. 41, No. 12, pp. 37-44, 1998.

Ropohl, G.: A Systems Theory of Technology (in German), Hanser, Munich, 1979.

Simpson, K.T.: The UI War Room and Design Prism: A User Interface Design Approach from Multiple Perspectives, in: User Interface Design, ed.: Wood, L.E., CRC Press, Boca Raton, pp. 245-274, 1998.

Stary, Ch.: Interactive Systems. Software Development and Software Ergonomics (in German), Vieweg, Braunschweig/Wiesbaden, 1996.

Stary, Ch.: The Role of Design and Evaluation Principles for User Interfaces for All, in: Proceedings HCI'97, pp. 477-480, 1997.

Stephanidis, C. (ed.); Salvendy, G.; Akoumianakis; Bevan, N.; Brewer, J.; Emiliani, P.L.; Galetsas, A.; Haataja, S.; Iakovidis, I.; Jacko, J.; Jenkins, P.; Karshmer, A.; Korn; P.; Marcus, A; Murphy, H.; Stary, Ch.; Vanderheiden, G.; Weber, G.; Ziegler, J.: Toward an Information Society for All: An International R&D Agenda, in: Human-Computer Interaction, Vol. 10, No. 2, pp. 107-134, 1998.

Volmerg, B.; Senghaas-Knoblock, E.: Technology Design and Responsibility (in German), Westdeutscher Verlag, Opladen, 1992.