UAIS (2001) 1: 40–55 / Digital Object Identifier (DOI) 10.1007/s102090100008

Universal Access in the Information Society: Methods, Tools, and Interaction Technologies

Constantine Stephanidis[1,2], **Anthony Savidis**[1]

[1] Institute of Computer Science, Foundation for Research and Technology – Hellas, Science and Technology Park of Crete, 71110, Heraklion, Crete, Greece; E-mail: cs@ics.forth.gr
[2] Department of Computer Science, University of Crete, Greece

Published online: 23 May 2001 –

Abstract. *Accessibility* and *high quality* of interaction with products, applications, and services by anyone, anywhere, and at any time are fundamental requirements for *universal access* in the emerging Information Society. This paper discusses these requirements, and their relation to the concept of automated adaptation of user interfaces. An example application is presented, showing how adaptation can be used to accommodate the requirements of different user categories and contexts of use. This application is then used as a vehicle for discussing a new engineering paradigm appropriate for the development of adaptation-based user interfaces. Finally, the paper investigates issues concerning the interaction technologies required for universal access.

Key words: Universal Access – Universal Design – User Interface for All – Unified User Interfaces – Adaption – Adaptability – Adaptivity – Non-visual interaction – Switch-based interaction

1 Introduction

The term "Universal Access" has several connotations. Some consider it a new, politically correct term referring to the introduction of "special features" for "special users" in the design of a product. For others, Universal Access elevates what designers call "good user-based design" to a broader concept targeted towards addressing the needs of all potential users [42]. Moreover, some believe that Universal Access has its historical roots in the US Communications Act of 1934, covering telephone, telegraph, and radio services, and aiming to ensure adequate facilities at reasonable charges, especially in rural areas, and to prevent discrimination on the basis of race, colour, religion, national origin, or sex [32]. For many others, the term is associated with the effort to provide and facilitate access to the built environment (e.g., buildings and landscapes) for people with functional disabilities [19]. In the early period, accessibility problems were primarily considered to concern only the field of Assistive Technology (AT), and consequently, accessibility entailed meeting prescribed requirements for the use of a product by people with disabilities [42, 44, 47]. Due to its perceived small size, the AT field exhibits a slow assimilation of technological change. For example, the transistor first became embedded into hearing aids many decades after its invention. With the advent of the digital computer, and its broad penetration in business activities, the accessibility issue re-appeared, as disabled and elderly people faced serious problems in accessing computing devices.

In the context of the emerging Information Society, Universal Access resurfaces as a critical quality target. This is due to the changing world view of disabled and elderly people[1], as well as to the pace of technological change, which in many cases delivers products and services requiring particular skills and abilities on the part of the human user (e.g., experience in the use of advanced technologies). In other words, as a result of recent technological developments (e.g., proliferation of diverse interaction platforms, such as wireless computing, wearable equipment, kiosks), the range of the population which may gradually be confronted with accessibility problems extends beyond the population of disabled and elderly users to include *all* people. Thus, today, Universal Access

N.B. The authors wish to acknowledge Prof. Reinhard Oppermann (reinhard.oppermann@gmd.de) from the Institute for Applied Information Technology of GMD, the National Research Centre for Information Technology in Germany, for acting as Managing Editor for this paper.

[1] Such changes have been brought about as a result of demographic pressures, the human rights movement, and national and international legislation.

refers to the global requirement of coping with diversity in: (i) the characteristics of the target user population (including people with disabilities); (ii) the scope and nature of tasks; and (iii) the different contexts of use and the effects of their proliferation into business and social endeavours.

In the development lifecycle of software products and services, the most demanding phase is probably the User Interface engineering process. More than a decade ago, software quality was mainly attributed to functional characteristics such as operational reliability, efficiency, and robustness. In the last decade, usability has become a prominent target, establishing interaction quality as another dimension of software quality. Today, software is continuously evolving to support human tasks in various new domains, and to facilitate operation from different situations of use. This progressive computerisation of everyday activities gave birth to the notion of anyone, any time, anywhere access [36]. In the context of HCI, Universal Access introduces a new perspective that recognises, respects, values, and attempts to accommodate a very wide range of human abilities, skills, requirements, and preferences in the design of computer-based products and operating environments. This eliminates the need for "special features" while at the same time fostering individualisation, and thus high quality of interaction and, ultimately, end-user acceptability.

Such a commitment should not be interpreted as a call for a single design solution suitable for all users, but instead as an effort to design products and services that can adapt themselves to suit the broadest possible end-user population. In doing this, the implication is that different solutions will be appropriate for different contexts of use, and that the developed user interfaces will be capable of automatically selecting and applying the most appropriate solution for each individual case.

This paper will present a concrete example of the contribution of adaptation-based techniques to the universal accessibility of software user interfaces. The paper is structured as follows. It begins by briefly reviewing the concept of adaptation, and the approaches to adaptation developed in the recent past, following a path that led from 'a posteriori' and 'ad hoc' adaptations to the use of automated adaptation mechanisms in user interfaces designed to be universally accessible (Sect. 2). Subsequently, it provides a concrete example of an application, namely the AVANTI web browser, where the user interface is capable of adapting to a variety of parameters, such as different user abilities, requirements, and preferences, and different contexts of use (Sect. 3). Finally, it discusses a design and engineering framework, namely the Unified User Interface Development methodology, comprising HCI design techniques, methods, architectural abstractions, and tools for the development of automatically adapting interfaces, and explains how such a methodology has been applied in the development of the AVANTI browser (Sect. 4).

The paper derives its argumentation from recent efforts in the context of European Commission funded collaborative research and development work (see the Acknowledgements) aiming to provide and demonstrate the feasibility of tools for building user- and usage-context-adapted interfaces accessible by different user groups, including disabled and elderly people [33, 35, 36, 38, 40, 43].

2 The need for automatic user interface adaptation in Universal Access

Adaptation is a key element for coping with diversity in the HCI field [8, 9, 15, 16, 22, 31, 34].

In this context, adaptation characterises software products that automatically modify (adapt) their interactive behaviour according to the individual attributes of users (e.g., mental/motor/sensory characteristics, preferences), and to the particular context of use (e.g., hardware and software platform, environment of use). Adaptation is a multi-faceted process, which can be analysed along three main axes, namely the *source* of adaptation knowledge, the *level* of interaction at which it is applied, and the *type* of information on which it is based [34] (see Fig. 1).

As far as the source of adaptation knowledge is concerned, one can identify two complementary classes: knowledge available at start-up, i.e., prior to the initiation of interaction (e.g., user profile, platform profile, usage context), and knowledge derived at run-time (e.g., through interaction monitoring, inspection of the computing environment). Adaptation behaviour based on the former type of knowledge is termed *adaptability* and reflects the ability of the interface to automatically tailor itself to the *initial* interaction requirements, as these are shaped by the information available to the interface. Adaptation behaviour based on the latter type of know-

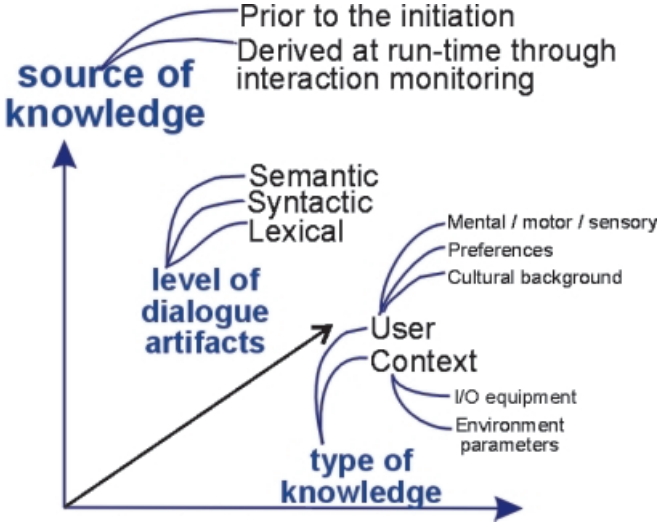

Fig. 1. Adaption dimensions in self-adapting interfaces

ledge is termed *adaptivity* and refers to the ability of the interface to dynamically derive further knowledge about the user, the usage context, etc., and to use that knowledge to further modify itself to better suit the revised interaction requirements.

The second axis of analysis of adaptation concerns the level of interaction at which adaptations are applied. In particular, it is possible to design and effect adaptations at all three levels of interaction:

(i) at the semantic level of interaction (e.g., by employing different metaphors to convey the functionality and facilities of the underlying system);
(ii) at the syntactic level of interaction (e.g., by de/activating alternative dialogue patterns, such as "object-function" versus "function-object" interaction sequencing); and
(iii) at the lexical level of interaction (e.g., grouping and spatial arrangement of interactive elements, modification of presentation attributes, alternative input/output devices).

The third main axis of analysis concerns the type of information being considered when deciding upon adaptations. Exemplary categories of information that can be employed include: design constraints, as these are defined by user characteristics (e.g., abilities, skills, requirements, preferences, expertise, cultural background), platform characteristics (e.g., terminal capabilities, input/output devices), task requirements (e.g., urgency, criticality, error-proneness, sequencing), etc. Furthermore, information that can be acquired only during interaction can equally participate in the decision process (e.g., identifying the user's inability to successfully complete a task, inferring the user's goal/intention, detecting modifications to the run-time environment).

Adaptable and adaptive software systems have been considered in a wide range of recent research efforts (e.g., [7, 12]). The relevant literature offers numerous examples illustrating tools for constructing adaptive interaction (e.g., [10, 11, 16, 18, 45]), and case studies in which adaptive interface technology has improved, or has the potential to improve, the usability of an interactive system (e.g. [3, 4, 12]). Until recently, however, adaptive techniques have had limited impact on the issue of Universal Access. In fact, in many cases, adaptive techniques and assistive technologies have shared terminological references (the most prominent being the concept of "adaptation" itself), sometimes with fundamental differences in the interpretations of these terms.

Early attempts to employ adaptation techniques to facilitate the accessibility of interactive software were motivated by the intention to serve specific user communities, such as disabled and elderly people[2]. Blind or visually impaired users attracted most of the attention in these early attempts. This was due to the particular challenges faced by this user community, resulting from the emergence of Graphical User Interfaces, which had severely limited their opportunities to access computers in comparison to the previous command-based interfaces that could more easily be rendered in a non-visual form. Adaptations in these early efforts were reactive in nature, in the sense that they sought to remedy the problem of inaccessibility once it had been introduced, rather than prevent it from occurring in the first place. The different approaches that emerged under this perspective can be broadly classified into two categories: product-level and environment-level adaptations. Product-level adaptations occur either as ad hoc modifications in already developed, inaccessible interactive products, or as dedicated product (re-)developments for particular categories of disabled people.

Irrespective of their intended purpose, these adaptations share a common characteristic: they are hard-coded into the application, and are static, in the sense that they are implemented as one-off accessibility solutions. Updating or modifying them to accommodate the slightest user variation or preference entails re-engineering the user interface. Although an in-depth discussion of these shortcomings is beyond the scope of this paper (interested readers can refer, for example, to [24, 25, 38]), it is important to note the following considerations. First of all, product-level adaptations introduce a programming-intensive approach towards accessibility, which increases the cost of implementing and maintaining accessible software. Secondly, technological progress may render such adaptations harder to implement; there may be restrictions imposed either by the target application or by the operating system. Thirdly, for practical and economic reasons, product-level adaptations always appear on the market with a considerable time lag behind the products they adapt.

Environment-level adaptations, on the other hand, moved away from the single-product level and addressed the "translation" of the visual, direct-manipulation dialogue realised by interactive software running within a software environment, to alternative modalities and media, accessible to disabled users. Several proposals emerged concerning the different types of tools that could be used to facilitate such adaptations. For example, in the case of blind users, the notions of "filtering" and "off-screen" models [21] were proposed as basic adaptation techniques facilitating access[3]. In a similar fashion, other types of software adaptation were developed for different categories of disabled users (e.g., motor impaired users, users with learning difficulties, etc.). Indicative examples include interface scanning, "sticky keys", and word

[2] Indicative projects addressing this research direction are the Mercator project in the USA [20], and a number of collaborative R&D projects funded by the European Commission [37].

[3] Progressively, these techniques have found their way into commercial products, which are today available in the market (e.g., screen readers).

prediction[4]. The major drawback of environment-level adaptations is rooted in the fact that they attempt to render accessible software that was designed for (and thus is inherently only appropriate for) "average" able-bodied users. No matter how advanced the adopted methods and techniques are, these types of adaptation are bound to far lower levels of interaction quality, when compared to interactive software specifically designed to cater for the particular needs and preferences of different categories of disabled users. These limitations become critically important when viewed in the context of the emerging Information Society, where accessibility can no longer be considered as a mere adaptation-based translation of interface manifestations to alternative modalities and media; it is instead a quality requirement demanding a more generic solution [42].

In the light of the above, it became evident that the challenge of accessibility needs to be addressed through more proactive and generic approaches, which account for all dimensions and sources of variation. These dimensions range from the characteristics and abilities of users, to the characteristics of technological platforms, to the relevant aspects of the context of use [42].

The concept of Dual User Interfaces[5] [24, 25] constituted a first step in this direction, since it defined a basis for "integrating" blind and sighted users in the same working environment. Dual User Interfaces signified a radical departure from previous approaches to user interface accessibility, by proposing that accessibility be treated from the early design phases of interactive software development, and that the accessibility requirements of more than one user category be taken into account. Thus the concept of Dual User Interfaces served as a catalyst towards proactive and more generic solutions to user interface accessibility, contributing new insights to the engineering of accessible user interface software. The basic premise of Dual User Interfaces, namely that accessibility can be tackled in a generic manner, was subsequently extended and further generalised through the concept of *User Interfaces for All* [33], and led to the development of a technical framework for supporting such an approach, as well as to the application of such a framework for the realisation of accessible and usable interactive applications.

The concept of *User Interfaces for All* is rooted in the idea of applying *Design for All* (or *Universal Design*, the terms are used synonymously) in the field of HCI [33, 36]. The underlying principle is to ensure accessibility at design time, and to meet the individual requirements of the user population at large, including disabled and elderly people. To this end, it is important that the needs of the broadest possible end-user population be taken into account from the early design phases of new products and services. Such an approach, therefore, eliminates the need for "a posteriori" adaptations and delivers interactive products that can be tailored, through automatic adaptability and adaptivity, to the individual requirements of all users. In the next section this approach is illustrated in practice, by showing how adaptation-based techniques are employed in the user interface of a web browser in order to achieve accessibility and high quality of interaction for a variety of target user groups in different contexts of use.

3 An adaptable and adaptive application: the AVANTI web browser

Having identified in the previous section automatic adaptation as a fundamental mechanism for universal access, this section reports on the main characteristics of a practical application, namely the AVANTI system[6], which was, to the authors' knowledge, the first to employ adaptive techniques in order to ensure accessibility and high-quality of interaction for all potential users[7]. AVANTI advocated a new approach to the development of web-based information systems [5, 6]. In particular, it put forward a conceptual framework for the construction of systems that support adaptability and adaptivity at both the content[8] and the user interface levels.

The user interface component of the AVANTI system is functionally equivalent to a web browser. In the AVANTI browser, user interface adaptability and adaptivity are applied to tailor the browser to the end-user abilities, requirements, and preferences, both during the initiation of a new session and throughout interaction with the system. The distinctive characteristic of the AVANTI browser is its ability to dynamically tailor itself to the abilities, skills, requirements, and preferences of the end-users, to the different contexts of use, and to the changing characteristics of users as they interact with the system [39, 40].

3.1 Adaptation requirements in the AVANTI web browser

The primary goal of adaptability in the AVANTI web browser is to ensure that each of the system's potential users is presented with an instance of the user interface that has been tailored to offer the highest possible degree

[4] Examples of Assistive Technology products for motor or cognitive impaired users can be found at the following URL addresses: http://www.olivetreesoftware.com/itmidx9.htm, http://www.donjohnston.com/, http://www.intellitools.com/, http://www.medizin.li/mt_index_az/_a/a_40034.htm, http://www.prentrom.com/speech/axs.html.

[5] The concept of Dual User Interface and the HOMER User Interface Management System have been developed in the context of GUIB and GUIB II projects (see Acknowledgements).

[6] The AVANTI information system has been developed in the framework of the AVANTI project (see Acknowledgments).

[7] The developed demonstrator addressed concurrently the requirements of able-bodied, blind, and motor-impaired users.

[8] For an extensive discussion of content-level adaptation in the AVANTI information system, see [14].

of accessibility (limited, of course, by the system's knowledge of the user). Adaptivity is subsequently employed to further tailor the system to the inferred needs or preferences of the user, so as to achieve the desired levels of interaction quality.

The AVANTI browser was designed to provide an accessible and usable interface to a range of user categories, irrespective of physical abilities or technology expertise. Moreover it was expected to support differing situations of use. The end-user groups targeted in AVANTI, in terms of physical abilities, include: (i) "able-bodied" people, assumed to have full use of all their sensory and motor communication "channels"; (ii) blind people; and (iii) motor-impaired people, with different forms of impairments in their upper limbs, causing different degrees of difficulty in employing traditional computer input devices. In particular, in the case of motor-impaired people, two coarse levels of impairment were taken into account: "light" motor impairments (i.e., users have limited use of their upper limbs but can operate traditional input devices or equivalents with adequate support) and "severe" motor impairments (i.e., users cannot operate traditional input devices at all).

Furthermore, since the AVANTI system was intended to be used both by professionals (e.g., travel agents) and by the general public (e.g., citizens, tourists), the users' experience in the use of, and interaction with, technology was another major parameter that was taken into account in the design of the user interface. Thus, in addition to the conventional requirement of supporting novice and experienced users of the system, two new requirements were put forward: (a) supporting users with any level of computer expertise; and (b) supporting users with or without previous experience in the use of web-based software.

In terms of usage context, the system was intended to be used both by individuals in their personal settings (e.g., home, office), and by the population at large through public information terminals (e.g., information kiosks at a railway station, airport). Furthermore, in the case of private use, the front end of AVANTI should be appropriate for general web browsing, allowing users to make use of the accessibility facilities beyond the context of a particular information system.

Additionally, users were to be continuously supported as their communication and interaction requirements changed over time, due to personal or environmental reasons (e.g., stress, tiredness, system configuration). This entailed the ability of the system to detect dynamic changes in the characteristics of the user and the context of use (of either a temporary or a permanent nature) and cater for these changes by appropriately modifying itself.

3.2 Adaptation scenarios from the AVANTI browser

In the AVANTI browser, adaptation is based on a number of 'static' and 'dynamic' user characteristics. 'Static' user characteristics are those that are unlikely to change in the course of a single interaction session (although they can change over longer periods of time). These characteristics are assumed to be known prior to the initiation of interaction (i.e., retrieved from the user profile). Static characteristics taken into account in the design of the AVANTI browser include: (a) physical abilities; (b) the language of the user (the demonstration system supports English, Finnish, Greek, and Italian); (c) familiarity of the user with: computing, networking, hypermedia applications, the web, and the AVANTI system itself; (d) the overall usage targets: speed, ease of use, accuracy, error tolerance; and (e) user preferences regarding specific aspects of the application and the interaction. The second set of characteristics is termed 'dynamic' to denote that the evidence they hold is usually derived at run-time, through interaction monitoring. The dynamic user characteristics (changing during interaction) and interaction states that are taken into account in AVANTI include: (a) familiarity with specific tasks (i.e., the user's ability to successfully initiate and complete certain tasks); (b) ability to navigate using the documents' navigation elements; (c) error rate; (d) disorientation; (e) user idle time; and (f) repetition of interaction patterns [40].

To illustrate some of the categories of adaptation available in the AVANTI browser, some instances of the browser's user interface will be briefly reviewed below.

Figure 2 contains two instances of the interface that demonstrate adaptation based on the characteristics of the user and the usage context. Specifically, Fig. 2a presents a simplified instance intended for use by a user unfamiliar with web browsing. Note the "minimalist" user interface with which the user is presented, as well as the fact that links are presented as buttons, arguably increasing their affordance (at least in terms of functionality) for users familiar with windowing applications in general. In the second instance, Fig. 2b, the interface has

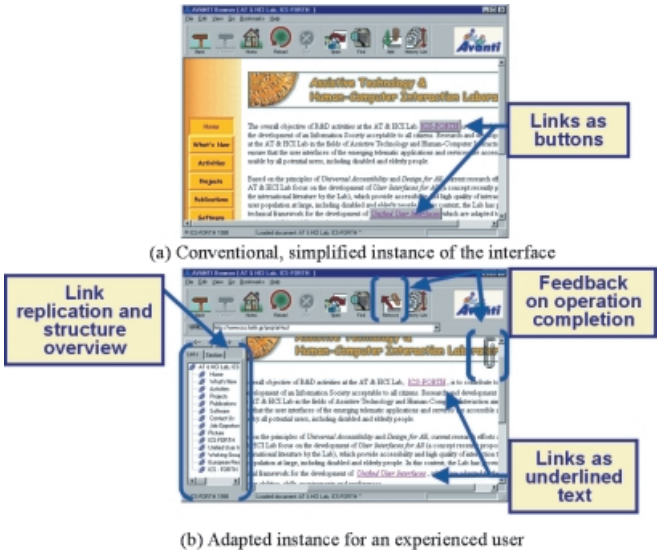

Fig. 2. Adapting to the user and the context of use

been adapted for an experienced user. Note the additional functionality that is available to the user (e.g., a pane where the user can access an overview of the document itself, or of the links contained therein, and an edit field for entering the URLs of local or remote HTML documents).

Figure 3 contains some sample instances demonstrating disability-oriented adaptations in the browser's interface. The instance in Fig. 3a presents the interface when a special interaction technique for motor-impaired users is activated, namely hierarchical interface scanning (either manually or automatically activated). Scanning is a mechanism allowing the user to "isolate" each interactive object in the interface and to interact with it through binary switches (see Sect. 4.3.2). Note the scanning highlighter over an image-link in the HTML document and the additional toolbar that was automatically added in the user interface. The latter is a "window manipulation" toolbar, containing three sets of controls enabling the user to perform typical actions on the browser's window (e.g., resizing and moving). Figure 3b illustrates the three sets of controls in the toolbar, as well as the "rotation" sequence between the sets (the three sets occupy the same space on the toolbar, to better utilise screen real estate, and to speed up interaction; the user can switch between them by selecting the first of the controls). Figure 3c presents an instance of the same interface with an on-screen "virtual" keyboard activated for text input. Interaction with the on-screen keyboard is also scanning-based.

The single interface instance in Fig. 4 illustrates a case of adaptive prompting [40]. Specifically, this particular instance is displayed in those cases in which there exists a high probability that the user is unable to initiate the "open location" task (this would be the case if there

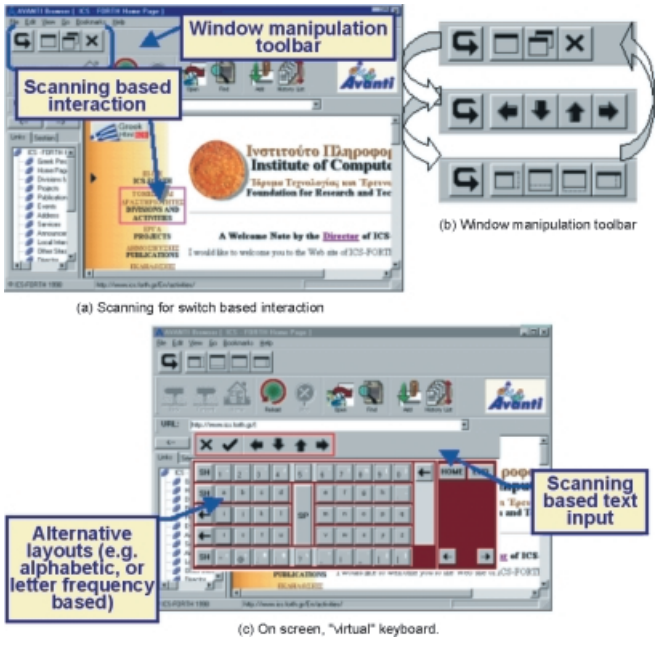

Fig. 3. Instances for motor-impaired users

Fig. 4. Awareness prompting

were adequate evidence that the user was attempting to load an external document with unsupported means, e.g., using 'drag and drop'). In this case, adaptive prompting is achieved through the activation of a "tip" dialogue, i.e., a dialogue notifying the user about the existence of the "open location" functionality and offering some preliminary indications of the steps involved in completing the task.

3.3 Key issues in developing user interface adaptation

The experience gained in the development of the AVANTI browser has demonstrated the effectiveness of adaptation-based techniques (both adaptability and adaptivity) when designing user interfaces which must be accessible and usable by a wide range of user categories, irrespective of physical abilities or technological expertise, and in a variety of contexts of use. The brief analysis presented in the previous section has demonstrated that the design space for the user interface of the AVANTI web browser was rather large, covering a range of diverse user requirements, different contexts of use, and dynamically changing interaction situations. The accessibility requirements posed by the target user groups addressed in the AVANTI development could not be met by existing browsers. Although today's commercially available browsers support customisability (e.g., through "add-on" components), the level of adaptation supported in the AVANTI browser can not be supported by customisation techniques (e.g., integrating guidance in system dialogues, dynamically modifying the interaction dialogue).

On the other hand, such requirements could not be met by traditional user interface development methods, since they lack design and implementation methods for providing adaptation mechanisms. Such methods should be capable of capturing the adaptation behaviour in the user interface design, and encapsulating it accordingly in the implementation, and should be supported by appropriate tools [37]. Since adaptation implies providing dialogue according to user- and context-related factors, suitable methods and tools should provide a means of capturing user and context characteristics and their interrelationships with alternative interactive behaviours,

as well as appropriate mechanisms for deciding adaptations on the basis of the those parameters and relationships. Additionally, suitable methods and tools should provide appropriate techniques for managing alternative implemented interactive behaviours and applying adaptation decisions at run-time. This is not the case in currently available interface development tools, which are mainly targeted to the provision of advanced support for implementing physical aspects of the interface via different techniques (e.g., visual construction, task-based, demonstration-based) [28].

Furthermore, commercial development toolkits do not provide support for interaction techniques which are alternatives to mouse / keyboard based graphical interaction. Clearly, the development of an application such as the AVANTI browser requires the availability of development toolkits capable of providing the high-quality interaction functionality required for both able-bodied and disabled users (e.g., in the case of AVANTI, motor-impaired and blind users). This is not the case in commercially available graphical environments, such as the widely available MS Windows.

The following section elaborates on these important aspects of adaptation by outlining the design and engineering methodology, as well as the interaction toolkits, employed in the development of the AVANTI web browser, focusing on those aspects that differentiate them from traditional HCI methods and make them a suitable framework for the development of universally accessible user interfaces.

4 An engineering paradigm for user interface adaptation: Unified User Interfaces

The Unified User Interface development methodology [2, 23, 26–28, 37] is a new methodology seeking to convey a new perspective on the development of user interfaces, and to provide a principled and systematic approach towards coping with diversity in the target user groups, tasks, and environments of use, through the use of automatic user interface adaptation[9].

Unified User Interface development entails an engineering perspective on interactive software, and can be performed through a collection of dedicated tools that allow the specification of a user interface as a composition of abstractions. A Unified User Interface comprises a single (unified) interface specification, targeted to potentially all user categories. A Unified User Interface is defined by the following key engineering properties: (a) it encompasses user- and context-specific information; (b) it contains alternative dialogue artifacts in an implemented form, each appropriate for different user- and context-specific parameters; and (c) it applies adaptation decisions, activates dialogue artifacts, and is capable of interaction monitoring.

The Unified User Interface development method comprises design-oriented techniques (Unified User Interface design) [23] aimed at the development of rationalised design spaces, and implementation-oriented techniques (Unified User Interface implementation) that provide a specifications-based framework for automatic user interface adaptation [2, 26, 28]. The rest of this section aims to briefly introduce the key characteristics of the Unified User Interface engineering paradigm, and outline how such a paradigm has been applied in the development of the AVANTI browser to support the adaptability and adaptivity behaviours described in Sect. 3.

4.1 Unified User Interface design

Unified User Interface design [23] aims to: (i) initially identify and enumerate possible design alternatives suitable for different users and contexts of use; (ii) identify abstractions and fuse alternatives into abstract design patterns; and (iii) rationalise the design space.

Enumeration of design alternatives is attained through techniques for analytical design (such as design scenarios, envisioning, ethnographic methods) which facilitate the identification of plausible design options for different user groups, computational platforms, environments and situations of use, etc.; the collection of all alternatives, at all levels of interaction, constitutes the design space of the interface.

Abstraction entails the identification of abstract interface components that are de-coupled from platform-, modality-, or metaphor-specific attributes. Abstractions are developed by developing a polymorphic task hierarchy [23]. The polymorphic task hierarchy is a construction in which the nodes represent abstract design elements, de-coupled from the specifics of the target user groups and the underlying interaction platforms, while the leaves depict concrete physical instances of the abstract design element suitable for specific contexts of use. In polymorphic task hierarchies, a task may be decomposed into an arbitrary number of alternative subhierarchies, thus depicting requirements or preferences of different user groups. Correspondingly, the leaf nodes represent specific physical instances of a design space generated through exploratory design efforts. Navigation down to the leaf nodes is guided by the association of design criteria that justify the differentiated artifacts with the specific requirements posed by individual users and contexts of use.

Finally, rationalisation of the design space implies the explicit encoding of the rationale for mapping an abstract design element to a concrete artifact. This is typically achieved by assigning criteria to design alternatives and providing a method for selecting the maximally preferred option, either at design- or at run-time.

[9] The Unified User Interface Development methodology has been developed in the framework of the ACCESS project (see Acknowledgments).

The result of the design process is a unified user interface specification. Such a specification can be realised using a dedicated, high-level programming language, and results in a single implemented artifact which can instantiate alternative patterns of behaviour, either at the physical, syntactic, or semantic level of interaction. Following the Unified User Interface Design method, in the design of the AVANTI browser, after defining the design space as well as the primary user tasks to be supported in the user interface, alternative dialogue patterns (referred to as *styles* in Unified Design) were defined/designed for each task to cater for the addressed user parameters. Consider, for example, the design of a web-browsing user task, namely that of specifying the URL of a web document to be retrieved and presented (this will be referred to as the "open location" task in the rest of this paper). This is a typical task in which user expertise is of paramount importance. For instance, an experienced user would probably prefer a "command-based" approach for the accomplishment of the task (which is the approach followed by most modern browsers), while a novice user might benefit from a simplified approach, such as choosing from a list of pre-selected destinations. When the physical abilities of the user are also considered, the "open location" task may need to be differentiated even further, in order to be accessible by a blind person, or a person with motor impairment. Moreover, there are cases in which the "open location" task should not be available at all, such as when the interface is used at a public information point.

During the unified design of the AVANTI user interface, the definition/design of alternative interaction styles was performed in parallel with the definition of the task hierarchy for each particular task. This reflected the iterative execution of the polymorphic task decomposition phase of the unified design method: the existence of alternative styles drives the decomposition process in sub-task hierarchies, and the task decomposition itself imposes new requirements for the definition of alternative styles for each defined sub-task. In practice, we identified those specific nodes within the task hierarchy for which polymorphic decomposition was required (due to the user and/or usage parameters), and produced alternative design artifacts to accommodate these requirements. In Fig. 5, the result of the polymorphic task decomposition process for the "open location" task is depicted.

The key factor which drives the polymorphic decomposition process is the user- and usage-context characteristics that impose certain requirements in the definition/selection of alternative styles and style components, and provide information and constraints on the physical appearance and interactive behaviour of the employed interaction objects. In the Unified Design method, two sets of user-oriented characteristics form the basis of the decision parameters driving the polymorphic decomposition process. These parameters concern static and dynamic user characteristics. Static characteristics typically comprise: sensory/motor/mental abilities, native language, cultural characteristics, age, familiarity with computing/web/the particular interactive application system itself, and particular user preferences regarding aspects of the application and the interaction. Dynamic characteristics may concern, for example, user familiarity with specific tasks (i.e., evidence of the user's ability to successfully initiate and complete certain tasks), error rate, and user idle time.

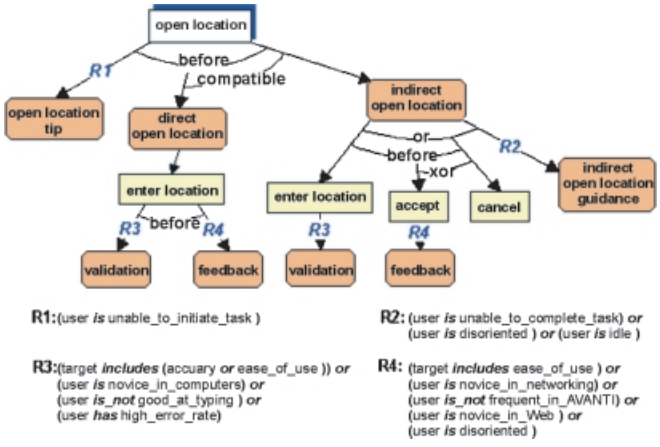

Fig. 5. An example of polymorphic hierarchical task decomposition

During the polymorphic task decomposition process, a set of user- and usage-context characteristics is associated with each polymorphic alternative during the decomposition process, providing the mechanism for deciding upon the need for, and selecting, different styles or style components. This set of characteristics constitutes the adaptation rationale, which is depicted explicitly on the hierarchical task structure in the form of design parameters associated with specific "values", which were then used as "transition attributes", qualifying each branch leading away from a polymorphic node. These attributes are later used to derive the run-time adaptation decision logic of the final interface.

4.2 Unified User Interface architecture

The Unified User Interface development process is complemented with an appropriate architectural framework [26], which promotes an insight into user interface software, based on structuring the implementation of interactive applications by means of independent intercommunicating components with well-defined roles and behaviours. In this architecture, the notion of encapsulation plays a key role: in order to realise system-driven adaptations, all the various parameters, decision-making logic, and alternative interface artefacts are explicitly represented in a computable form, constituting integral parts of the run-time environment of an interactive system.

A Unified User Interface performs run-time adaptation to provide different interface instances for different end-users and usage-contexts. Figure 6 depicts

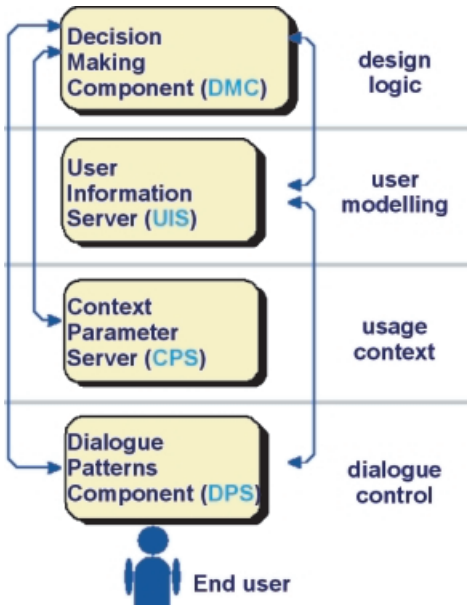

Fig. 6. The architecture of a Unified User Interface

the basic components of the Unified User Interface architecture [26].

These are:

(a) The Dialogue Patterns Component (DPC). This module implements all the alternative dialogue patterns identified during the design process, on the basis of various user and context attribute values. The DPC may employ pre-developed interactive software, in combination with additional interactive components. The latter case is normally expected if the directly deployed interactive software does not provide all the required implemented patterns addressing the target user- and usage-context attribute values. The DPC should be able to apply pattern activation/cancellation decisions originating from the Decision Making Component. Additionally, interaction-monitoring components may be attached to various implemented dialogue patterns, providing monitoring information to the User Information Server for further processing (e.g., key-strokes, notifications for use of interaction objects, task-level monitoring).

(b) The User Information Server (UIS). This module maintains the individual profiles of end-users, which, together with usage-context information (provided by the Context Parameters Server), constitute the primary input source for the adaptation decision-making process (delegated to the Decision Making Component). From a knowledge representation point of view, static or pre-existing user knowledge may be encoded in any appropriate form, depending on the type of information the UIS should propagate into the decision making process. Moreover, additional knowledge-based components may be employed for processing retrieved user profiles, drawing assumptions about the user, or even updating the original user profiles. Apart from such initial (i.e., prior to initiation of interaction) manipulation of user profiles, the UIS may also collect and process run-time interaction events, in order to draw (additional) inferences about the end-user. Such inferences may result in the identification of dynamic user preferences, loss of orientation in performing certain tasks, fatigue, inability to complete a task, etc.

(c) The Context Parameters Server (CPS). This component encompasses information regarding the usage environment and interaction-relevant machine parameters. During the interface design process, the identification of those important parameters relevant to the context(s) of use, will need to be carried out. This module is intended to provide device awareness, thus enabling the Decision Making Component to select those interaction patterns which, apart from fitting the particular end-user attributes, are also appropriate for the type of equipment available on the end-user machine. The usage-context attribute values are communicated to the Decision Making Component before the initiation of interaction; additionally, during interaction, some dynamically changing usage-context parameters may also be communicated to the Decision Making Component for decisions regarding adaptive behaviour.

(d) The Decision Making Component (DMC). This module encompasses the logic for deciding the necessary adaptation actions, on the basis of the user and context attribute values, received from the UIS and CPS respectively. Such attribute values will be supplied to the DMC prior to the initiation of interaction (i.e., initial values, resulting in initial interface adaptation, referred to as adaptability) as well as during interaction (i.e., changes in particular values, resulting in dynamic interface adaptations, referred to as adaptivity). The DMC is responsible only for deciding the necessary adaptation actions, which are then directly communicated to, and subsequently performed by, the DPC. The run-time adaptation process in Unified User Interfaces is practically a context-sensitive selection of those components which have been designed to address that particular situation. In order to perform such a context sensitive selection, the DMC encompasses information regarding all the various dialogue patterns present within the DPC, and their specific design roles. There are two categories of adaptation actions which are decided on and communicated to the DPC: (i) activation of specific dialogue components; and (ii) cancellation of previously activated dialogue components.

The Unified User Interface architecture has been intentionally developed to be easily embedded in existing interactive software frameworks, by requiring only some

additional implemented modules, mainly serving coordination, control, and communication purposes. This capability facilitates the vertical growth of existing interactive applications, following the Unified User Interface architectural paradigm, so that automatically adapted interaction can be accomplished.

The development of the AVANTI browser has been based on the Unified User Interface architectural framework. The rest of this section briefly elaborates on the mapping of the various implementation components of the AVANTI browser (Fig. 7) to the Unified User Interface architectural components (Fig. 6).

Firstly, the adaptable and adaptive interface components, the interaction monitoring component, and the page presentation and interaction component in the AVANTI browser architecture are the functional equivalent of the Dialogue Patterns Component (DPC) in the Unified User Interface architecture. They encapsulate the implementation of the various alternative dialogue patterns (interaction/instantiation styles) identified during the design process, and are responsible for their activation/de-activation, applying the adaptation decisions made by the respective module. Moreover, each style implementation has integrated functionality for monitoring user interaction and reporting the interaction sequence back to the user modelling component of the AVANTI system.

Secondly, the adaptation mechanism directly corresponds to the Decision Making Component (DMC) in the Unified User Interface architecture. It encompasses the logic for deciding on and triggering adaptations, on the basis of information stored in its knowledge space (this information is, in turn, retrieved from user profiles and inferences based on interaction monitoring). Furthermore, adaptations can be triggered both during the initiation of interaction and at run-time.

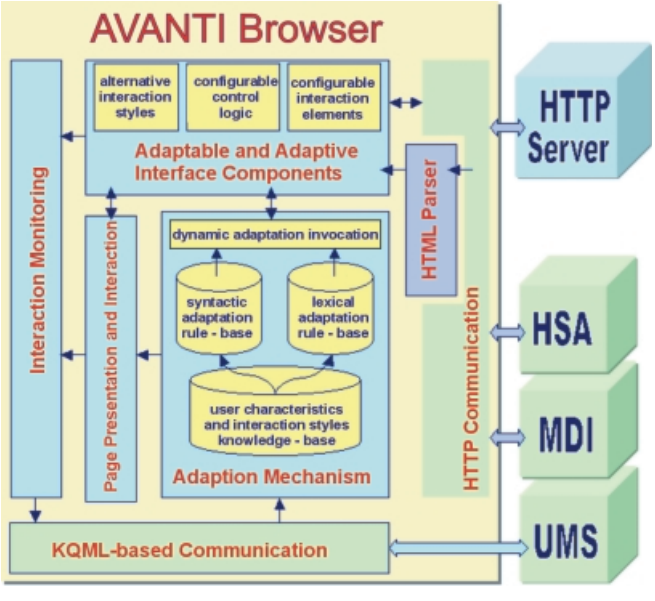

Fig. 7. The AVANTI web browser architecture as an instance of the Unified User Interface architecture

The role of the User Information Server (UIS) in the Unified User Interface architecture is played by the UMS of the AVANTI system [18]. The UMS is implemented as a server, usually remotely located on the network, since it offers central, multi-user modelling functionality. The communication between the user interface and the UMS is bilateral: the interface sends messages regarding user actions at the physical and task levels (e.g., "the user pressed reload", or "the user successfully completed the loading of a new document", respectively). The UMS employs a set of stereotypes that store categorised knowledge about the users and their interactions and environment, and a set of rules to draw inferences on the current state of the user, based on the monitoring information.

Finally, the Context Parameters Server (CPS) in the Unified User Interface architecture does not have an equivalent component in the AVANTI implementation, since context-specific adaptation was not addressed by the AVANTI project.

4.3 Unified User Interface tools

The implementation of Unified User Interfaces is supported by a development environment that includes the USE-IT Design Support Tool [1, 2] and the I-GET User Interface Management System [28].

The USE-IT Design Support Tool provides the design-time support for achieving adaptability. Specifically, USE-IT decides upon the lexical aspects of the interaction dialogue based on knowledge of user characteristics, abilities, and preferences, as well as knowledge of the structure of lexical level characteristics of the interface with respect to the various target user groups.

I-GET is a User Interface Management System [46] providing all the necessary mechanisms for implementing Unified User Interfaces. It supports the high-level specification of Unified User Interfaces, as well as the automatic generation of user interface implementation from such specifications.

The I-GET language for interface implementation exhibits a number of powerful new features, when compared to existing development tools, including:

- *Toolkit integration*: I-GET is capable of importing virtually any toolkit, relying upon a Generic Toolkit Metamodel, the Toolkit Interface Specification language kernel, and the specifically designed Generic Toolkit Interfacing Protocol for communication.
- *Toolkit abstraction*: I-GET takes a step beyond the desktop metaphor and provides openness and extensibility of virtual objects, by supporting the definition of arbitrary virtual objects, and the specification of the physical instantiation logic, through which alternative ways of mapping abstract objects to physical objects can be defined.

- *Toolkit augmentation*: I-GET supports the enhancement (augmentation) of an existing toolkit defined as the process of implementing additional interaction techniques, as part of the particular metaphor realisation originally implemented by the toolkit, in order to enable interaction for particular user categories. Newly introduced interaction techniques become part of the augmented toolkit software library, while all interactive applications built with the augmented toolkit automatically support the new features of enhanced interaction.
- *Toolkit expansion*: I-GET supports a process through which users of a toolkit (i.e., user interface developers) can introduce new interaction objects not originally supported by that toolkit
- Facilities for manipulating diverse dialogue patterns: I-GET offers a syntax-loose dialogue control method based on component classes called "agents" (exhibiting hierarchical organisation and precondition- or call-based instantiation), combined with powerful constructs such as monitors, preconditions, and constraints for arbitrary variables.

Furthermore, the I-GET UIMS integrates toolkits supporting the implementation of interaction techniques which are alternatives to direct manipulation, such as the HAWK toolkit for non-visual interaction [29] and the SCANLIB toolkit for switch-based interaction [30].

Since extensive accounts of the I-GET and USE-IT tools are provided elsewhere [1, 2, 28], the rest of this section concentrates on the interaction technologies integrated in the Unified User Interface development platform, and exploited in the development of the AVANTI browser to provide viable alternatives to direct manipulation for certain target user groups and contexts.

4.3.1 Support for non-visual interaction

The HAWK toolkit [29] provides a set of standard non-visual interaction objects and interaction techniques that have been specifically designed to support high quality non-visual interaction. Such a toolkit is not only appropriate for interfaces targeted to blind users, but also for a variety of situations in which dialogues not relying on visual communication and standard input devices are required (e.g., driving, telephone-based applications, home control auditory interaction). This is achieved through the provision of interaction primitives that do not impose a specific metaphor, but enable alternative metaphoric representations to be built. In this respect, HAWK differs from windowing toolkits that provide real-world analogies of physical objects (such as menus, buttons, switches, potentiometers).

A key notion in this context is that of "container interaction object". In the HAWK toolkit there is a single generic container class that does not provide any predesigned interaction metaphor, but supplies appropriate presentation attributes through which alternative representations can be created.

The container class has four attributes that enables each distinct container instance to be given a metaphoric substance by appropriately combining messages and sound-feedback (both names and sound effects can have a metaphoric meaning, see Fig. 8).

In addition to generic containers, HAWK provides a comprehensive collection of conventional interaction objects directly supporting non-visual dialogue, namely menus (exclusive choice selector object), lists (multiple choice selector object), buttons (push button analogy for direct command execution), toggles (radio button analogy for on/off state control, edit fields (single line text input) and text reviewers (multi-line read-only text editor, with mark-up facilities).

HAWK also supports a variety of interaction techniques, namely synthesised speech, Braille (2-cell transitory Braille, 40-cell Braille), and digitised audio for output; and standard keyboard, joystick used for gestures independently of visual interaction, touch-tablet (for programmable commands via associated regions), and voice recognition for input.

The HAWK toolkit provides all the programming features found in currently available toolkits, such as hierarchical object composition, dynamic instantiation, callback registration, and event handling. The navigation dialogue enables the blind user to move within the interface structure composed of organisations of containers and contained objects in a simple way, through the provision of multi-modal control facilities. For instance, visiting contained objects is possible through joystick-based gestures, voice commands, keyboard short cuts, or via pressing specific regions of the touch-tablet. Container objects may contain other container objects realising different metaphoric representations, thus supporting fusion of different metaphors in the context of non-visual interactive applications.

The efficiency and effectiveness of the navigation dialogue within the object hierarchies of the developed User Interfaces is an important factor for the acceptance of an

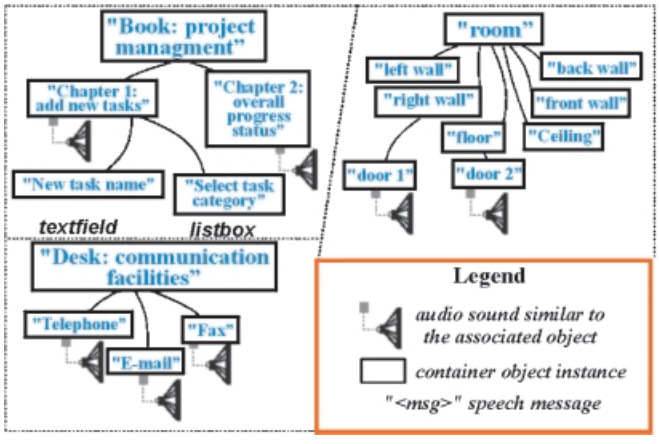

Fig. 8. Supporting different metaphoric representations for container instances

interface toolkit. In visual windowing environments, the direct perception of the physical realisation of interaction object hierarchies is enabled, thus sighted users may directly review various interface components. In non-visual interaction the provision of powerful navigation facilities is much more critical, as it is much more difficult to enable the direct perception of the overall interface structure. The HAWK toolkit provides a flexible architecture for implementing navigation dialogues configurable according to target user needs. Navigation dialogues are split into two parts: (i) input channel and input commands (see left part of Fig. 9); and (ii) navigation commands (see middle-right part of Fig. 9). Binding input commands to navigation commands defines the navigation dialogue (for instance, defining that the up-left joystick gesture maps to the "Previous object" navigation command), and mapping schemes can be freely defined by end-users.

The specific input commands of the input channels supported by the HAWK toolkit can also be reprogrammed (with the exception of joystick gestures, which are fixed). For instance, keyboard shortcuts, speech keywords, and touch-tablet regions can be redefined according to user and application requirements.

Navigation commands are handled by container objects, while the input channels register user input in the case of non-container objects, which receive input commands in the form of typical toolkit input events (programmers will implement event handlers). If the requirements of the navigation dialogue cannot be met through the utilisation of the configuration facilities, alternative approaches can be taken. For example, the navigation dialogue can be disabled (disabling can be specialised even at the object instance level for containers), and as a result, the HAWK toolkit will provide conventional input event notification for input commands to those container objects which have the navigation dialogue disabled. Alternatively, the HAWK toolkit library for navigation commands, which makes all navigation commands available as software functions, can be used to implement the desired dialogue in the form of event handlers for those input commands needed. For instance, the built-in joystick gestures can be substituted by implementing event handlers that recognise new gestures.

The main advantage of the HAWK toolkit is its independence of specific metaphors. This feature enables interface designers to structure interaction in a way reflecting the non-visual mental representation of blind users. In this respect, the HAWK toolkit builds upon and generalises the COMONKIT toolkit [24, 25], based on a non-visual version of the Rooms metaphor.

In the development of the AVANTI browser, the HAWK toolkit was used to provide a set of appropriate input and output devices, along with appropriate interaction techniques, to visually impaired users. Non-visual input and output in AVANTI can involve any of the following devices/systems: keyboard (or any keyboard emulation device); mouse/trackball (or any mouse emulation device); touch screen; Braille display; touch tablet; binary switches; joystick; speech synthesis (output) and speech/command recognition (input); non-speech audio output. Touch tablets can be used by blind users through demarcated areas (raised edges, Braille labels, etc.), each of which corresponds to specific functionality. Speech synthesis is used to present textual information to blind users, and to signify attributes related to the possible hypermedia nature of the presented documents (e.g. links), while speech recognition can be used to allow blind users to issue vocal commands to the system, through a special set of control and navigation commands. Gesture recognition permits the use of a joystick by blind users, by coupling specific gestures to command sequences. Finally, tactile presentation of hypertext in Braille is augmented with special symbolic annotations that facilitate the user's comprehension of the exact type of item being presented.

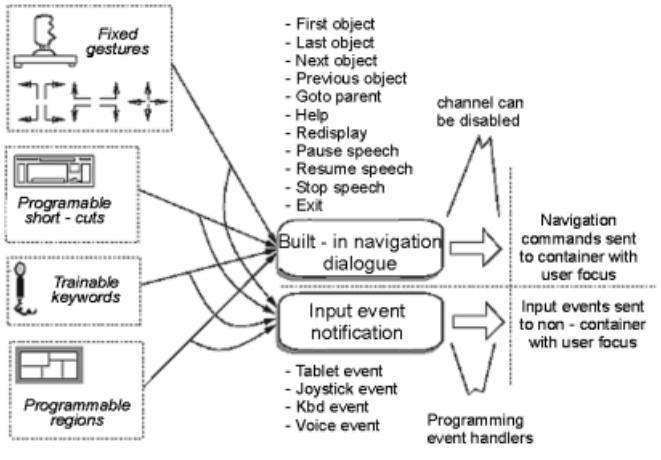

Fig. 9. The four concurrent input channels for navigation dialogue and conventional user input in the HAWK toolkit

4.3.2 Support for switch-based interaction

In existing graphical environments, such as the widely available WINDOWS 98/2000/NT, the ability to support switch-based interaction is rather limited. Previous approaches, suitable for WINDOWS 3.1, relied on the support of windows-based operations via key bindings [48, 49]. Users had to explicitly program such key bindings, on the basis of scripting methods, to create an extra interface on top of WINDOWS (accessible via scanning techniques) for artificial generation of WINDOWS 3.1 key shortcuts. Moreover, it was impossible to differentiate behavioural attributes of the scanning interaction technique for different user tasks (e.g. to have various scanning highlighters depending on the interaction context). These approaches present considerable operational limitations under recent WINDOWS versions, due to the significant architectural changes. As a consequence, it is necessary that the full range of graphical interaction

techniques in a visual windowing environment be appropriately augmented to support interaction via such special-purpose peripherals. The main advantage of augmented toolkits is maximum control of extra interaction facilities by application developers, while the main disadvantage is that old applications cannot benefit from the extensions, unless they are rebuilt and appropriately extended on the basis of the new toolkit library.

In the SCANLIB interface development toolkit [30], the basic WINDOWS 95 object library has been augmented with scanning interaction techniques. Interfaces implemented through this augmented software library directly support motor-impaired user access, as well as access in other situations in which the keyboard and mouse input devices can not be used. Apart from enabling intra-application interaction control (e.g., having access via switches to all interface elements of any interactive application), SCANLIB also supports inter-application interaction control (e.g., enabling users to move across different applications). This requires low-level shared memory facilities, so that the augmented library can always be aware of the number of augmented applications currently running. In SCANLIB, basic object classes are classified into five categories, each requiring a different dialogue policy to be designed:

- Top-level windows: these are the objects directly providing window management operations. Since in WINDOWS 95 it is impossible to "access" (in a programmable manner) the built-in controls (icons) for window management, all top-level windows in SCANLIB have been augmented with an additional toolbar accessible via scanning.
- Container objects: these are object classes with instances present as intermediate nodes in object hierarchies, able to encompass an arbitrary number of contained object instances. Container objects enable sequential scanning of all contained objects.
- Text-entry objects: objects requiring text to be supplied impose the need for keyboard emulation. When the user focuses (via scanning) on a particular text-field object, a special on-screen keyboard automatically appears, through which text input is enabled (Fig. 4). Such an augmented dialogue is realised transparently to programmers, to whom the augmented text-field object at the programming level appears as a conventional WINDOWS text-field object.
- Composite objects: typical examples of composite objects are the scroll-bar class (composed of two arrow buttons and a slider) and the combo-box class (composed of a drop-down button, a label, and a menu object). Composite objects enable sequential scanning of all component-objects.
- Button categories: these are simple interaction objects supporting direct user actions for executing associated operations or changing the state of boolean parameters. Typical examples from this category are: push buttons, check boxes, and radio buttons.

The scanning interaction techniques are based on two fundamental actions: SELECT and NEXT. Upon entering a container object, such as a group-box that may enclose an arbitrary number of objects (e.g., other group-boxes, list-boxes, radio-buttons), the dialogue starts in "exit mode". This is indicated to the user via a special highlighting. If the user initiates the "select" action (i.e., presses the "select" switch) when in exit mode, the dialogue will move to the next object in the hierarchy, thus skipping the group-box and all enclosed objects. If the user initiates the "next" action, the dialogue shifts to the "entry" mode, and feedback is provided by changing highlighting style. By initiating a "select" at this state, the user will start a dialogue with the (hierarchically) first object of the group-box, while by initiating a "next" action, the dialogue state again changes to "exit" mode (thus, successive "next" actions will cycle between "exit" and "entry" modes).

Depending on the type of switch equipment required, four scanning modes are supported:

- 1-switch (for SELECT action), time scanning (automatic NEXT action, after a specific time interval).
- 2-switches (one for SELECT action, one for changing scanning direction), time scanning (as before for automatic NEXT action).
- 2-switches (SELECT action, NEXT action), no time scanning.
- 5-switches (SELECT action, NEXT action, change direction, directly exit container, restart with container).

Programming control is provided for the extra attributes introduced by the augmented version of WINDOWS 95 objects, thus enabling objects to differentiate scanning style depending on scanning mode (one of the five alternatives), time interval (when time scanning is supported), and highlighter presentation parameters. By default, contained objects directly inherit all the scanning attributes from the direct container parent object, unless explicit values are supplied.

SCANLIB, as a development toolkit, can be used to create new applications completely accessible through switch-based interaction, although it can not be used to add scanning facilities to existing applications. One of its main advantages is the provision of window management facilities providing full-control to motor-impaired users.

In the development of the AVANTI browser, the SCANLIB toolkit has been used to support switch-based interaction for motor-impaired users through automatic or user-controlled scanning and on-screen keyboards (see Fig. 3c).

5 Discussion and Conclusions

In this paper the Unified User Interface concept has been introduced, showing that user- and usage-context self-adapting interfaces are a promising technical vehicle for achieving Universal Access. The paper elaborated on

the Unified User Interface design method and the Unified User Interface engineering paradigm, putting forward a profile of a new technology for developing universally accessible interactions. Additionally, the paper described special purpose toolkits for building interactions beyond the traditional desktop interaction style (typically based on keyboard and mouse), and discussed their employment in the context of a Unified development process for building a specific application, notably the AVANTI adaptable and adaptive web browser.

There are a number of issues that emerge from this paper. Recent R&D activities have revealed valuable insights into the study of Universal Access in HCI, in particular concerning the contribution of automatic adaptation [34, 38]. As accessibility and interaction quality have become global requirements in the Information Society, adaptation needs to be "designed into" the system rather than decided upon and implemented "a posteriori". Therefore, at the level of design methods and techniques, Universal Access requires an understanding of the global execution context of a task. This entails design techniques that can capture alternative design options and design representations that can be incrementally extended (i.e., design pluralism) to encapsulate evolving or new artifacts. Another important issue concerning design, which has been learned through practice and experience, is that Universal Access means breaking away from the traditional perspective of "typical" users interacting with a desktop machine in a business environment and moving towards interactive systems accessible at any time, anywhere and by anyone. This means that future systems should embody the capability for context-sensitive processing so as to present their users with a suitable computational embodiment or metaphor depending on user, situational, and context-specific attributes.

The Unified User Interface development methodology is the first systematic effort in this direction that has been thoroughly developed and applied in practice. The results of such an effort and the experience gained justify the argument that developing universally accessible and usable user interfaces is more of a challenge than a utopia [38]. However, Unified User Interfaces should not be considered as the only possible approach for addressing Universal Access in the Information Society. Although Unified User Interface development assumes powerful tools and advanced user interface software technology, such a realisation is not the only possible, or superior to any alternative that may be developed in the future. Nevertheless it is argued that, in the long run, the availability of such tools is likely to be a critical factor for the potential adoption and diffusion of the proposed concept. The Unified User Interface development framework augments and enhances traditional approaches to developing user interfaces in the following dimensions: (i) it is oriented towards iteratively capturing and satisfying multiple user- and context-oriented requirements; (ii) it employs adaptation-based techniques in order to facilitate and enhance personalised interaction. Furthermore, Unified User Interface development is characterised by the need to identify appropriate evaluation methods and processes to assess the effectiveness of the self-adapting behaviour of the interface.

The development of the AVANTI browser has constituted an instantiation of the Unified User Interface development process. The AVANTI unified interface is capable of adapting itself to suit the requirements of able-bodied, blind, and motor-impaired users in a variety of contexts of use. Adaptability and adaptivity are used extensively to tailor and enhance the interface respectively, in order to effectively and efficiently meet the target of interface individualisation for end users. Since the design of the user interface has followed the Unified User Interface design method, and the implementation is based on the Unified User Interface architecture, inclusion of additional target user groups and contexts of use has been effectively facilitated [26]. The experience acquired has demonstrated the applicability of the methodologies, techniques, and tools comprising the Unified User Interface development paradigm in the construction of a large-scale "real-world" self-adapting interactive application.

The Unified User Interface development process is a resource-demanding design and engineering process. It requires the systematic classification and organisation of both design and engineering artifacts, according to unified design and engineering practices, thus exhibiting a relatively high entry barrier for development. Multidisciplinary expertise is required, especially within the implementation phase. For instance, decision-making will typically require logic-programming methods, while management of user information may combine database management (for user profiles) with user model representation methods (e.g., declarative programming and knowledge bases). The experience gained so far has clearly indicated the considerable development overhead of crafting Unified User Interfaces, an overhead that is proportional to the targeted degree of diversity in users and usage-contexts. Additionally, it has been observed that an initial "start-up" investment is required to set up a unified user interface architecture, starting from the preliminary stages prior to the actual embedding of components in a running system, e.g., implemented dialogues, decision-making logic, or user information models. Finally, an additional need has emerged for an automated integration process for expanding the system's adaptation behaviour to address additional user or context parameter values, and subsequently to inject implemented decision-making logic and dialogue patterns. In this context, future work is targeted towards the production of reusable implemented architectural patterns [17], in the form of configurable application frameworks [13], which will accompany the Unified User Interface engineering paradigm, and will help to quickly instantiate and easily expand a Unified User Interface implementation.

Concerning development, an important observation is that prevailing graphical user interface toolkits do not suffice to facilitate the broad range of alternative interactive embodiments needed to empower and facilitate Universal Access. In this respect, traditional user interface architectural models may need to be revised and extended to inform and guide developments in this direction, and appropriate tools are needed to facilitate the development of universally accessible and usable systems. In particular, the suitability of such tools depends on their independence of specific development platforms. Tools considered appropriate are those that instead of directly calling a platform can link to it and make use of the available interaction resources. As a derivative of the previous argument, it follows that user interface development should progressively be supported through specification-based rather than programming-oriented techniques.

Despite the recent rise of interest in the topic of Universal Access, and the successful and indisputable progress in R&D, many challenges still lie ahead. Firstly, an effective use of adaptation in user interfaces implies a deep and thorough knowledge of the user requirements. This implies identifying diversity in the user population, mapping diversity-related requirements to concrete designs, and evaluating the related adaptation behaviour. The same holds for the study of context which, as pointed out in previous sections, is critical for the quality of a system's adaptable and adaptive behaviour [41].

The above remark introduces another challenge to be addressed, namely the compelling need to assess the implications of Universal Access for digital contents, functionality, and interaction. This entails, amongst other things, a renewed account of the properties of adaptation and how they can be embodied by digital contents, functionality, and the user interface. Clearly, the traditional view that adaptable and adaptive behaviour is a characteristic property of the interactive software may no longer suffice given the trends towards ubiquitous access, mobile computing, and Internet appliances.

Finally, theoretical work has to be supported by large-scale case studies, which can provide the instruments for experimentation, thus ultimately improving the empirical basis of the field. Such case studies should aim not only to demonstrate technical feasibility of R&D propositions, but also to assess the economic efficiency and efficacy of competing technological options in the longer term.

Acknowledgements.

- The GUIB TP103 (Textual and Graphical User Interfaces for Blind People) project was partially funded by the TIDE Programme of the European Commission, and lasted 18 months (December 1, 1991 to May 31, 1993). The partners of the GUIB consortium are: IROE-CNR, Italy (Prime Contractor); F H Papenmeier GmbH & Co KG (Germany); IFI-University of Stuttgart (Germany); Institute of Computer Science-FORTH (Greece); RNIB (England); Institute of Telecommunications-TUB (Germany); Department of Computer Science-FUB (Germany); Vrije Universiteit Brussels (Belgium); VTT (Finland).
- The GUIB-II TP215 (Textual and Graphical User Interfaces for Blind People) project was partially funded by the TIDE Programme of the European Commission, and lasted 18 months (June 1, 1993 to November 30, 1994). The partners of the GUIB-II consortium are: IROE-CNR (Italy); Institute of Computer Science-FORTH (Greece); Vrije Universiteit Brussels (Belgium); Department of Computer Science-FUB (Germany); Institute of Telecommunications-TUB (Germany); IFI, University of Stuttgart (Germany); VTT (Finland); RNIB (England); F.H. Papenmeier Gmb & Co KG (Germany).
- The ACCESS TP1001 (Development platform for unified ACCESS to enabling environments) project was partially funded by the TIDE Programme of the European Commission, and lasted 36 months (January 1, 1994 to December 31, 1996). The partners of the ACCESS consortium are: CNR-IROE, Italy (Prime Contractor); ICS-FORTH (Greece); University of Hertfordshire (United Kingdom); University of Athens (Greece); NAWH (Finland); VTT (Finland); Hereward College (United Kingdom); RNIB (United Kingdom); Seleco (Italy); MA Systems & Control (United Kingdom); PIKOMED (Finland).
- The AVANTI AC042 (Adaptable and Adaptive Interaction in Multimedia Telecommunications Applications) project was partially funded by the ACTS Program of the European Commission, and lasted 36 months (September 1, 1995 to August 31, 1998). The partners of the AVANTI consortium are: ALCATEL Italia, Siette division, Italy (Prime Contractor); IROE-CNR (Italy); ICS-FORTH (Greece); GMD (Germany); VTT (Finland); University of Siena (Italy); MA Systems and Control (UK); ECG (Italy); MATHEMA (Italy); University of Linz (Austria); EUROGICIEL (France); TELECOM (Italy); TECO (Italy); ADR Study (Italy).
- The authors wish to thank the anonymous reviewers for their valuable comments.

References

1. Akoumianakis D, Stephanidis C (1997) Supporting user-adapted interface design: the USE-IT System. Interact with Comput 9(1):73–104
2. Akoumianakis D, Stephanidis C (2001) USE-IT: a tool for lexical design assistance. In [35], pp 469–487
3. Benyon DR (1993) Adaptive systems: a solution to usability problems. User Modelling User-adapted Interact 3(1):1–22
4. Benyon DR (1997) Intelligent interface technology to improve human-computer interaction. Tutorial in HCI International '97. San Francisco, USA
5. Bini A, Emiliani PL (1997) Information about mobility issues: the ACTS AVANTI project. In Proceedings of 4th European Conference for the Advancement of Assistive Technology (AAATE '97), Porto Carras, Greece. IOS Press, Amsterdam, pp 85–88
6. Bini A, Ravaglia R, Rella L (1997) Adapted interactions for multimedia based telecommunications applications. In Conference Neties 1997, University of Ancona, Ancona, Italy
7. Browne D (1993) Experiences from the AID project. In [31], pp 69–78
8. Browne D, Totterdell P, Norman M (eds) (1990) Adaptive user interfaces. Academic Press, UK
9. Brusilovsky P (1996) Methods and techniques of adaptive hypermedia. User Modelling User-Adapted Interact 6(2–3):87–129
10. Brusilovsky P, Kobsa A, Vassileva J (eds) (1998) Adaptive hypertext and hypermedia systems. Kluwer Academic Publishers, Dordrecht, Netherlands
11. Cote-Munoz AH (1993) AIDA: an adaptive system for interactive drafting and CAD applications. In [31], pp 225–240
12. Dieterich H, Malinowski U, Kühme T, Schneider-Hufschmidt M (1993) State of the art in adaptive user interfaces. In [31], pp 13–48

13. Fayad M, Cline M (1996) Aspects of software adaptability. Commun ACM 39(10):58-59
14. Fink J, Kobsa A, Nill A (1998) Adaptable and adaptive information provision for all users, including disabled and elderly people. New Rev Hypermedia Multimedia 4:163–188
15. Hayes-Roth B, Pfleger K, Lalanda P, Morignot P, Balabanovic M (1995) A domain-specific software architecture for adaptive intelligent systems. IEEE Trans Software Eng 21(4): 288–301
16. Horvitz E, Breese J, Heckerman D, Hovel D, Rommelse K (1998) The lumiere project: bayesian user modeling for inferring the goals and needs of software users. In Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, pp 256–265
17. Jacobson I, Griss M, Johnson P (1997) Making the reuse business work. IEEE Comput 30(10):36–42
18. Kobsa A, Pohl W (1995) The user modelling shell system BGP-MS. User Modelling User-adapted Interact 4(2):59–106
19. Mace RL, Hardie GJ, Plaice JP (1991) Accessible environments: toward universal design. In Preiser W, Vischer J, White E (eds) Design Interventions: Toward a more Human Architecture. Van Nostrand Reinhold, New York
20. Mynatt ED, Edwards WK (1992) The mercator environment: a nonvisual interface to the X window system. Graphics Visualization and Usability Center, Technical Report GIT-GVU-92-05
21. Mynatt ED, Weber G (1994) Nonvisual presentation of graphical user interfaces: contrasting two approaches. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '94). ACM Press, New York, pp 166–172
22. Oppermann R (ed) (1994) Adaptive user support: ergonomic design of manually and automatically adaptable software. Lawrence Erlbaum Associates, Hillsdale, NJ
23. Savidis A, Akoumianakis D, Stephanidis C (2001) The Unified User Interface design method. In [35], pp 417–440
24. Savidis A, Stephanidis C (1995) Developing dual user interfaces for integrating blind and sighted users: the HOMER UIMS. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '95). ACM Press, New York, pp 106–113
25. Savidis A, Stephanidis C (1998) The HOMER UIMS for dual user interface development: fusing visual and non-visual interactions. Int J Interact Comput 11(2):173–209
26. Savidis A, Stephanidis C (2001a) The Unified User Interface software architecture. In [35], pp 389–415
27. Savidis A, Stephanidis C (2001b) Development requirements for implementing Unified User Interfaces. In [35], pp 441–468
28. Savidis A, Stephanidis C (2001c) The I-GET UIMS for Unified User Interface implementation. In [35], pp 489–523
29. Savidis A, Stergiou A, Stephanidis C (1997) Generic containers for metaphor fusion in non-visual interaction: the HAWK interface toolkit. In Proceedings of the Interfaces '97 Conference, Montpellier, France, 28–30 May. Montpellier District & EC2 Development, pp 194–196
30. Savidis A, Vernardos G, Stephanidis C (1997) Embedding scanning techniques accessible to motor-impaired users in the WINDOWS Object Library. In Salvendy G, Smith MJ, Koubek RJ (eds) Design of Computing Systems: Cognitive Considerations [Proceedings of the 7th International Conference on Human-Computer Interaction (HCI International '97), San Francisco, USA], vol 1. Elsevier, Elsevier Science, Amsterdam, pp 429–432
31. Schneider-Hufschmidt M, Kühme T, Malinowski U (eds) (1993) Adaptive user interfaces: principles and practice. North-Holland, Elsevier Science, Amsterdam
32. Shneiderman B (2000) Universal usability: pushing human-computer interaction research to empower every citizen. CS-TR-4043. Commun ACM 43(5):84–91
33. Stephanidis C (1995) Towards user interfaces for all: some critical issues. Panel session "User Interfaces for All – Everybody, Everywhere, and Anytime" In Anzai Y, Ogawa K, Mori H (eds) Symbiosis of human and artifact – future computing and design for human-computer interaction [Proceedings of the 6th International Conference on Human-Computer Interaction (HCI International '95), Tokyo, Japan]. Elsevier, Elsevier Science, Amsterdam, pp 137–142
34. Stephanidis C (2001) Adaptive techniques for Universal Access. User Modelling User Adapted Interact Int J, 10th Anniversary Issue, 11(1–2):159–179
35. Stephanidis C (ed) (2001a) User interfaces for all – concepts, methods and tools. Lawrence Erlbaum Associates, Mahwah, NJ. ISBN 0-8058-2967-9
36. Stephanidis C (2001b) User interfaces for all: new perspectives into HCI. In [35], pp 3–17
37. Stephanidis C (2001c) The concept of Unified User Interfaces. In [35], pp 371–388
38. Stephanidis C, Emiliani PL (1999) Connecting to the Information Society: a European perspective. Technol Disability J 10 (1):21–44 [On-line]. Available at: http://www.ics.forth.gr/proj/at-hci/files/TDJ_paper.PDF
39. Stephanidis C, Paramythis A, Sfyrakis M, Savidis A (2001) A case study in Unified User Interface development: the AVANTI web browser. In [35], pp 525–568
40. Stephanidis C, Paramythis A, Sfyrakis M, Stergiou A, Maou N, Leventis A, Paparoulis G, Karagiannidis C (1998a) Adaptable and adaptive user interfaces for disabled users in the AVANTI Project. In Trigila S, Mullery A, Campolargo M, Vanderstraeten H, Mampaey M (eds) Proceedings of the 5th International Conference on Intelligence in Services and Networks (IS&N '98), Technology for Ubiquitous Telecommunication Services. Lecture Notes in Computer Science, Vol. 1430. Springer, Heidelberg, Germany, pp 153–166
41. Stephanidis C, Salvendy G, Akoumianakis D, Arnold A, Bevan N, Dardailler D, Emiliani PL, Iakovidis I, Jenkins P, Karshmer A, Korn P, Marcus A, Murphy H, Oppermann C, Stary C, Tamura H, Tscheligi M, Ueda H, Weber G, Ziegler J (1999) Toward an information society for all: HCI challenges and R&D recommendations. Int J Hum Comput Interact 11(1):1–28
42. Stephanidis C, Salvendy G, Akoumianakis D, Bevan N, Brewer J, Emiliani PL, Galetsas A, Haataja S, Iakovidis I, Jacko J, Jenkins P, Karshmer A, Korn P, Marcus A, Murphy H, Stary C, Vanderheiden G, Weber G, Ziegler J (1998b) Toward an information society for all: an international R&D agenda. Int J Hum Comput Interact 10(2):107–134 [On-line]. Available at: http://www.ics.forth.gr/proj/at-hci/files/white_paper_1998.pdf
43. Stephanidis C, Savidis A, Akoumianakis D (1997) Unified interface development: tools for constructing accessible and usable user interfaces. Tutorial no. 13 in the 7th International Conference on Human-Computer Interaction (HCI International '97), San Francisco, USA, 23–29 August 1997. [On-line]. Available at: http://www.ics.forth.gr/proj/at-hci/html/tutorials.html
44. Story MF (1998) Maximising usability: the principles of universal design. Assistive Technol 10:4–12
45. Sukaviriya P, Foley J (1993) Supporting adaptive interfaces in a knowledge-based user interface environment. In Gray WD, Hefley WW, Murray D (eds) Proceedings of the International Workshop on Intelligent User Interfaces. ACM Press, New York, pp 107–114
46. UIMS Developers Workshop (1992) A meta-model for the runtime architecture of an interactive system. SIGCHI Bulletin 24(1):32–37
47. Vanderheiden G (1998) Universal design and assistive technology in communication and information technologies: alternatives or compliments? Assistive Technol 10(1):29–36
48. WinSCAN (1997) WinSCAN Version 2.0. http://www.acsw.com/ws1.html
49. WiVik (1997) http://www.prentrom.com/access/wivik.html.