Universal accessibility in HCI: Process-oriented design guidelines and tool requirements

C. Stephanidis, D. Akoumianakis, M. Sfyrakis, and A. Paramythis
Institute of Computer Science (ICS)
Foundation for Research and Technology-Hellas (FORTH)
Science and Technology Park of Crete
Heraklion, Crete
GR-71110, GREECE
Tel.: + 30 - 81 - 391741, Fax : +30 - 81 - 391740
email: cs@ics.forth.gr

Abstract. This paper presents a preliminary collection of design-oriented guidelines and development requirements for accessibility and universal design in HCI. The process-oriented guidelines aim to shed light into how a user-centred design process can be conducted, so as to account for the needs and requirements of the broadest possible end user population, including people with disabilities. These guidelines are subsequently translated into key development requirements which should be preserved in user interface development tools in order for them to provide the required support for building user interface software for different users and contexts of use. To this effect, we provide contextual definitions of key terms of reference and an account of related standards. The proposed material does not intend to cover a particular technology. Instead, it aims to formulate a conceptual framework whereby accessibility becomes an integral component of the user interface development life-cycle.

1. Introduction

The vast majority of the available accessibility guidelines are formulated either as general design principles, or low level and platform specific recommendations. They are typically based on past experiences and best practice, while experimental evidence is typically rare [Casali, 1995]. Additionally, they cover specific user groups, such as blind and motor-impaired users, and provide guidance on how user interface software can be adapted to become accessible [Bergman and Johnson, 1995]. Thus, for example, guidelines for WWW accessibility mainly focus on three aspects, namely *page authoring*, *user agents*, and the *structure and presentation* of Web documents. By implication, such guidelines do not fully address structural languages (e.g. XML), scripting languages and other properties which are typically related to the overall interaction platform. On the other hand, the proliferation of interaction platforms and their continuous growth (e.g., HTML, VRML, XML, DHTML), necessitate an account of key requirements that should be preserved, if these developments and future ones are to comply with the broad accessibility objectives. Moreover, such guidelines offer limited guidance on the process of integrating accessibility into design and development activities.

Though the value of such guidelines is beyond doubt, this paper adopts a slightly different normative perspective. In particular, it aims to: (a) provide process-oriented guidance, through guidelines, on accessibility and universal design in HCI; and (b) translate the resulting guidelines into requirements that need to be preserved by user interface development tools in order for them to provide the required support for building user interface software accessible

by the broadest possible end user population, including people with disabilities. The scope of the process-oriented design guidelines and the corresponding development requirements is deliberately broad in an attempt to provide a conceptual framework, independent of a particular technology, whereby universal accessibility is integrated in the user interface development life-cycle.

2. Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of the present document. At the time of writing, the editions indicated were valid. All standards are subject to revisions.

- ⇨ Draft International Standard (DIS) 13407, Human Centred Design process for interactive systems. International Standards Organisation, Geneva, Switzerland, 1997.
- ⇨ International Standard 9001, Quality Systems - Model for Quality Assurance in Design, Development, Production, Installation and Servicing. International Standards Organisation, Geneva, Switzerland, 1987.
- ⇨ Draft International Standard (DIS) 8402, *Quality Vocabulary*. International Standards Organisation, Geneva, Switzerland, 1994.
- ⇨ Draft ISO/IEC 14581-1*: Information Technology Evaluation of Software products-General guide.*
- ⇨ Draft International Standard (DIS) 9241-11, *Ergonomic Requirements for office work with visual display terminals, Part 11: Guidance on Usability*, International Standards Organisation, Geneva, Switzerland, 1997.
- ⇨ Draft International Standard (DIS) 14915 - Multimedia User Interface Design; Software Ergonomic Requirements

In addition, relevant documents containing accessibility guidelines are the Draft HFES/ANSI 200, Section 5: Accessibility (http://web.ansi.org), as well as the World Wide Web Consortium (W3C) Accessibility Guidelines (http://www.w3.org/WAI). Finally, W3C-WAI (Web Accessibility Initiative) pursues standardisation activities in the area of accessibility guidelines.

3. Definitions

This section provides definitions of some of the key terms, as well as contextual clarification, that are needed to facilitate a better understanding of the range and scope of the material to be presented. For terms which have already been defined in the relevant literature, we provide the corresponding definition together with any relevant comments that may be applicable. For other terms, which are not reserved, we provide some contextual clarification.

Universal design

The term *universal design* or *design for all* (the two terms are used interchangeably in this paper) is frequently associated with different connotations [Story, 1998]. Some individuals consider it as a new, politically correct, term, referring to efforts to introduce “special

features" for "special users" in the design of a product. To others, universal design is a deeply meaningful and rich topic that elevates what designers like to call "good user-based design" to a more encompassing concept of addressing the needs of all potential users.

In this paper (see also [Stephanidis et al, 1998]):

> *the term is used to reflect a new concept, or philosophy for HCI design that recognises, respects, values and attempts to accommodate the broadest possible range of human abilities, requirements and preferences in the design of all computer-based products and environments. Thus, it promotes a design perspective that eliminates the need for "special features" and fosters individualisation and end-user acceptability. As already pointed out, the term is used interchangeably with the term design for all users. This does not imply a single design solution suitable for all users. Instead, it should be interpreted as an effort to design products and services, in such a way, so as to suit the broadest possible end user population. In doing this, it is more than likely that there will be different solutions for different contexts of use.*

Universal accessibility

Accessibility is traditionally associated with disabled and elderly people and reflects the efforts devoted to the task of meeting prescribed code requirements for use by people with disabilities [Bergman and Johnson, 1995; Story, 1998]. However, due to recent technological developments (e.g., proliferation of interaction platforms, such as wireless computing, wearable equipment, user terminals), the range of the population which may gradually be confronted with accessibility problems extends beyond the population of disabled and elderly users.

In this paper (see also [Stephanidis et al, 1998]):

> *accessibility implies the global requirement for access to information by individuals with different abilities, requirements and preferences, in a variety of contexts of use; the meaning of the term is intentionally broad to encompass accessibility challenges as posed by diversity in: (i) the target user population profile (including people with special needs) and their individual and cultural differences; (ii) the scope and nature of tasks (especially as related to the shift from business tasks to communication and collaboration intensive computer-mediated human activities); (iii) the technological platforms and associated devices through which information is accessed.*

Usability, context of use and quality in use

In the recent literature, there are several definitions of the term usability. In this paper, we adopt the definition offered in ISO DIS 9241-11, though such a definition offers a performance-oriented perspective on usability.

Thus, in this paper (see also [ISO, 1997a]):

> *usability is defined as the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.*

The context of use is defined in [ISO, 1997a] as:

the nature of the users, tasks and physical and social environments in which a product is used.

Finally, the notion of quality is typically associated with various meanings and connotations (see also [Garvin, 1984], [Bevan, 1997]), while there is also variation with regards to how it can be achieved as part of the production process (e.g., [ISO, 1987]; [ISO, 1994]).

In this paper (see also [Stephanidis et al, 1998]):

quality in use covers both functional and non-functional[1] attributes which determine computer-mediated human activities.

User-centred design

The term user-centred design appeared in the relevant literature as early as 1985 in the title of a book edited by Norman and Draper [Norman and Draper, 1986] and which was aiming to advance new perspectives in human computer interaction. More recently, the term Human Centred Design was introduced to denote a multi-disciplinary activity, which incorporates human factors and ergonomics knowledge and techniques, and advances an approach to interactive system development that focuses specifically on making systems usable [ISO, 1997b].

In this paper:

the term is associated with a collection of attitudes, approaches and design processes through which users are directly involved / consulted throughout an iterative system development process.

It should be noted that the emphasis in the above definitions is on the process of design rather than the tools that may be used to conduct specific design activities. Thus, for example, the definitions do not distinguish between participatory (e.g., users are directly involved) versus non-participatory (e.g., users may be replaced by descriptions or models, such as in the case of GOMS) means to design.

Interaction platform

The term interaction platform refers to

any software tool providing implemented (or the means to implement) interaction elements which in turn can be used to construct a user interface.

Such software tools include the traditional user interface development toolkits, such as Windows95™, Motif, Athena Widget Set, as well as some current and emerging Web technologies such as structural languages (e.g., HTML, XML), presentation languages (e.g., CSS), scripting languages (e.g., JavaScript) as well as emerging Web technologies (such as WebTV, Java, etc).

[1] Examples of non-function quality attributes include portability, reusability, performance, modifiability, scalability, etc.

4. Universal accessibility principles, guidelines and recommendations

4.1 Terms

The recent literature on accessibility and universal design, provides several collections of general design principles, guidelines and recommendations for building accessibility into interactive computer-based products (see Figure 1). Design principles refer to high level design objectives which realise the notion of accessibility. Principles give rise to guidelines which may relate either to the syntactic- or physical-level of interaction [Akoumianakis and Stephanidis, 1998]. Guidelines are typically subject to further interpretation so as to reflect the requirements of a particular organisation or design case. Finally, recommendations are unambiguous statements about physical artefacts[2].

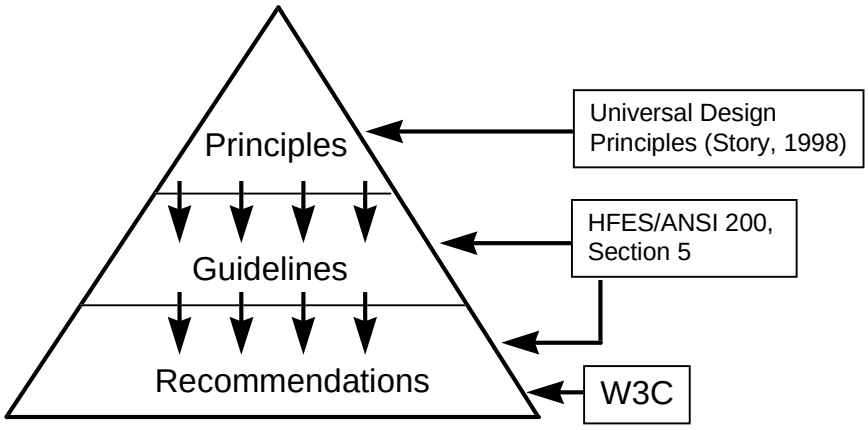

Figure 1: Conceptual structure of accessibility requirements

To illustrate the contextual difference of the above terms, let us consider the simple scenario described in Exhibit 1.

Exhibit 1

A user is to carry out a text editing task. The user has mild motor impairments which delimit control to gross temporal movements, exercised through contact with the fist. Fingertips cannot be reliably employed due to tremor on key-press, while movements can be performed in timed patterns and upon demand. The user is to carry out a text editing task.

Figure 2 depicts how a designer may progressively identify, select and interpret accessibility principles and guidelines into concrete recommendations. Thus, for instance, the software ergonomic principle which requires that: "the user interface should enable the user to initiate and accomplish a task" may translate to the two guidelines depicted in Figure 2, which in turn, give rise to recommendations of the physical "character" of the interface.

However, neither the software ergonomic principle, nor the derived guidelines and recommendations offer any guidance to the designer as to how he or she should proceed to identify and select appropriate interface components. Moreover, there is also no account of

[2] By physical artefact, we imply an interaction element (e.g., an interaction object or a composite component, such as a dialogue) bound to a particular toolkit (e.g., Windows95™, Motif).

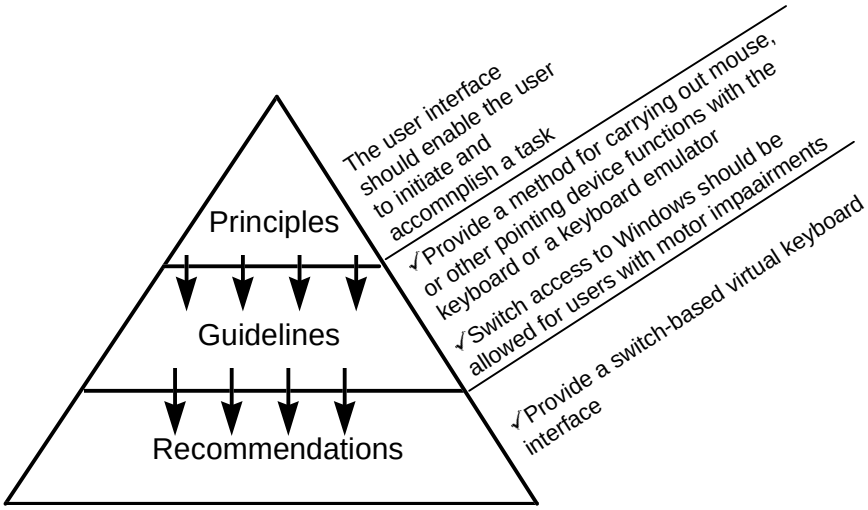

Figure 2: Progressively identifying, selecting and interpreting principles and guidelines into concrete recommendations.

any special features that the development tools should possess in order to realise a particular design.

To address the above, the present document presents: (a) three process-oriented design guidelines that extend user-centred design towards universal accessibility; and then (b) derives four development tool requirements for constructing user interfaces which are accessible by the broadest possible end user population, including people with disabilities. The intention is to extend the accumulated wisdom on accessibility and universal design in HCI, by focusing on the conduct of design, and by identifying required and recommended technical properties of user interface development environments.

4.2 Scope

Our interest is not to suggest specific (alternative) interaction techniques to support accessibility by different user categories, including disabled and elderly people. Instead, we will be concerned with process-oriented design guidelines and corresponding development requirements which result in the development of user interface software accessible by the broadest possible end user population. The process-oriented guidelines are intended for designers and identify steps to be followed within a human-centred life cycle. On the other hand, the derived development requirements provide guidance to developers as to the required or recommended features that should be observed when making a selection of user interface development environment. It is expected that these contributions augment the accumulated accessibility wisdom beyond mere adaptations, towards universal design practice.

4.3 Process-Oriented Design Guidelines & Development Requirements

One important observation related to the accessibility of interactive applications and services by different user groups, including people with disabilities, is that no single interface implementation is likely to suffice for all different users. This simple observation leads to the conclusion that *designing for the broadest possible end user population requires the provision of alternative interface manifestations depending on the abilities, requirements and preferences of the target user groups* [Stephanidis et al, 1998]. To attain the above target requires a human-centred design activity which has three additional key requirements, namely:

⇨ *enumeration* of design alternatives;

⇨ *encapsulation* of design alternatives into abstractions; and

⇨ *rationalisation* of the resulting design space.

Figure 3 depicts these steps and reveals techniques that may be used to accomplish each one.

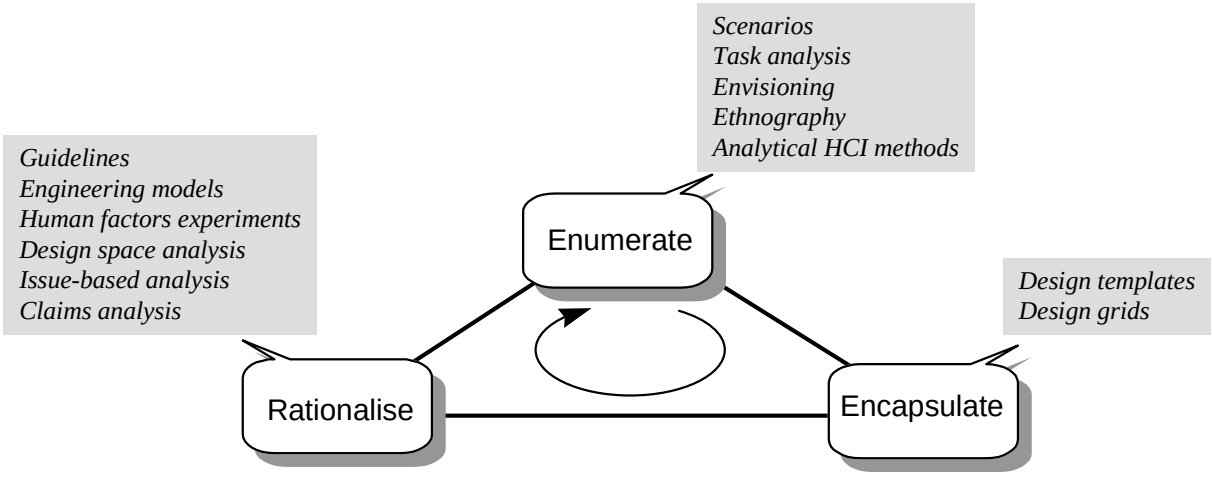

Figure 3: Requirements for designing for the broadest possible end user population

4.3.1 Designers should seek to identify and enumerate plausible design alternatives suitable for the target user groups

Design alternatives are necessitated by the different contexts of use (see definition above) and provide a global view of task execution. This is to say that design alternatives offer rich insight into how a particular task may be accomplished by different users in different contexts of use.

Since users differ with regards to abilities, requirements and preferences, tentative designs should aim to accommodate the broadest possible range of capabilities across different contexts of use. Thus, instead of restricting the design activity to producing a single outcome, designers should strive to compile design spaces containing plausible alternatives.

As an example, consider the primitive interaction task of selection. A selection may by made either by choosing an option from a menu (see options [a] and [c] in Figure 4) or by issuing a command (option [b]), etc. Moreover, as illustrated by options [a] and [c] in Figure 4, the menu may be conveyed in different design languages[3]. For example, the use of the word "menu" in option [a] is borrowed from the "restaurant" domain of discourse; the command in option [b] follows the typewriter's metaphor; the circular clock in option [c], resembles the operation of an electric device (e.g., a potentiometer).

[3] A design language is defined as a mechanism mediating the mapping of concepts in a source domain (e.g., restaurant, typewriter, electric device) to symbols in a presentation domain (e.g., interaction elements offered by a particular toolkit).

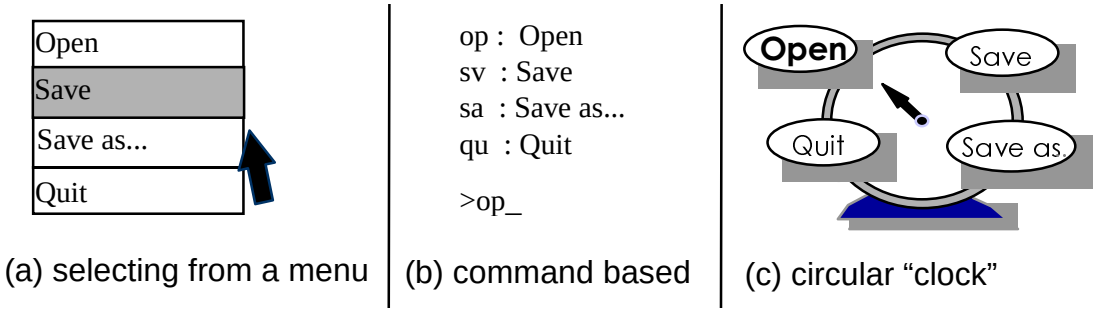

Figure 4: Alternative embodiments of selection in different design languages

Figure 5: Non-visual alternatives for selection

What is important to note, however, is that none of the above alternatives, or any other visual option that one may come up with, would be considered suitable for a blind user who lacks the capabilities to attain information conveyed in the visual modality. Instead, blind people would be more comfortable using alternative manifestations conveyed either in audio, or tactile modalities (see Figure 5).

4.3.2 Plausible design alternatives should be encapsulated into extensible design abstractions

Once the design space has been compiled and documented, the design activity should proceed towards the encapsulation of plausible alternatives into abstract, reusable and extensible design components. The need for abstraction is two-fold. First of all, abstractions may be used to de-couple a design concept from any particular physical realisation which is tied to a specific interaction platform; thus, through abstractions, a design concept may be mapped onto alternative physical counterparts. Secondly, abstractions provide a mechanism to support incremental design and design updates, as requirements mature, or evolve; thus, the original design space may be extended to include new physical realisations, necessitated by new contexts of use or made possible by novel interaction technologies.

Consider, for example the options presented in Figure 4 and Figure 5. Removing the presentation-specific details which differentiate one option from another, one arrives at two attributes that constitute an abstract selector, namely numberOfOptions and userChoice. Such an abstraction could be mapped to any of the options depicted in Figure 4 and Figure 5; additionally, it may be mapped to any other design that may be identified in the future either through refinement of an existing alternative, or through evolution.

4.3.3 Designers should rationalise the design space by exemplifying the logic for mapping design abstractions onto concrete user interface artefacts

Rationalisation of the design space entails a principled approach to defining the reasoning behind each alternative. This enables the designer to map design abstractions to physical counterparts and provide the evidence for the mapping (see Figure 6).

In order to produce the necessary evidence, designers may have to assess alternatives with end users, or carry out experimentation to develop comparative results and decide on maximally preferred options. Thus, for example, a comparative test may be set up to assess the circumstances and the parameters determining the choice between option [a] and option [c] from Figure 4, for making a selection. Such a comparative test may be carried out using alternative user-centred design techniques, including subjective user assessments, performance measurement, GOMS analysis, etc.

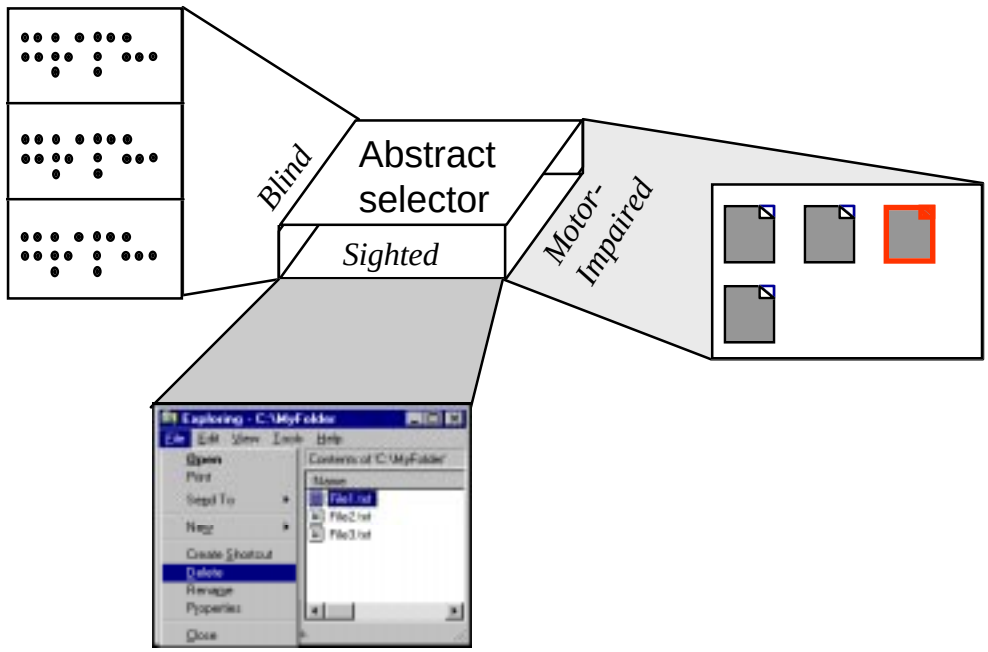

Figure 6: Mapping abstractions to alternative physical counterparts

The need for rationalisation necessitates a shift from artefact-oriented design towards process-oriented and analytical design, whereby the reasoning behind an artefact is equally important as the artefact itself.

4.4 Requirements for development tools

Having reviewed the process-oriented guidelines for universal design in HCI, this section aims to derive key requirements for user interface development tools. In order to achieve this, we will briefly elaborate the implications of the design guidelines upon user interface development.

First of all, enumeration implies a conscious effort to populate design spaces (e.g., design pluralism) by extrapolating plausible usage scenarios and encountering artefacts suitable for different contexts of use. Moreover, many of these artefacts may be conveyed in different design languages and alternative modalities. Iterative prototyping of these artefacts demands

the availability of tools to create versions which will enable users to experience and comment on the envisioned artefacts. Through such iterations, initial sketches become progressively high fidelity prototypes.

What is important to note is that a single interaction platform will most likely not suffice to accommodate all plausible alternatives. Thus, developers will be confronted with the requirement to select and integrate different platforms (e.g., visual and non-visual toolkits). However, it may also be the case that due to novel design properties, a specific platform (e.g., toolkit) may need to be extended to support additional interactive behaviours.

Secondly, encapsulation of alternatives into abstractions requires that developers need constructs to map abstract components to concrete interaction elements in a manner that is intuitive and relieved from the specifics (e.g., programming model) of a particular interaction platform. This, in turn, necessitates a shift towards specification of interactive behaviours rather than programming.

Finally, rationalisation implies the capability for context-sensitive processing of alternatives and the selection of the maximally preferred option, given a context of use. Moreover, as the context of use can not be fully known a priori, developers should be provided with suitable Application Programming Interfaces (APIs) that enable inter-operability amongst the user interface software and external components that may hold context-related knowledge.

The above translate to several tool requirements which need to be supported in order to provide the grounds for constructing universally accessible user interfaces. These requirements can be summarised as follows:

A development tool should provide facilities for:

- ✓ importing interaction resources offered by different interaction platforms *(platform integration)*
- ✓ augmenting the originally supported interaction techniques with new ones, suitable for specific users and contexts of use (*platform augmentation)*
- ✓ specifying interactive behaviours through abstract interaction elements relieved from platform-specific properties *(platform abstraction)*
- ✓ exposing and making use of information produced by external software tools (*orthogonality*)

In the following sections, we review each one of those requirements and discuss required and recommended features that development tools should possess to meet them.

4.4.1 Integration

Platform integration entails the capability to import any interaction platform that may be required for the development of interactive applications, so that all interaction elements of the imported platform can be directly accounted by the same interaction building techniques. Platform integration is necessitated in cases where the interaction elements originally supported by a particular interaction platform do not suffice to provide support for a particular type of interaction (e.g., non-visual). In such cases, it is important for the development tools to

be able to utilise interaction elements from alternative sources (e.g., external object libraries), offering the interaction facilities required.

It is important to note that, usually, interaction building blocks and re-usable interface components which are provided from different software firms do not follow interoperability guidelines, thus introducing several impediments to platform integration.

To support the platform integration requirement, a development tool should possess several required features. These are:

(a) ability to link / mix code at the software library level;

(b) documented hooks provided by platform developers, in order to allow mixing at the source code level.

Recommended features include:

(i) single implementation model made available for all integrated platforms, irrespective of the style of interaction supported;

(ii) ability to modify aspects of the programming interface (i.e., the programmable view) of each imported platform;

(iii) inter-operability of the integrated platforms.

4.4.2 Augmentation

Platform augmentation refers to the process through which additional interaction techniques are injected within the original collection of interaction elements of a particular platform. The rationale for platform augmentation arises from the fact that it is sometimes desirable to provide extended interaction facilities, beyond the original collection, which could be useful in specific contexts of use (e.g., voice-control of windowing application, scanning). In augmented development platforms, newly introduced interaction techniques should become an integral part of original toolkit elements, while old applications re-compiled with the augmented toolkit version automatically inherit the extra interaction features.

For a development tool to support platform augmentation, it is required that:

(a) new devices are installed;

(b) programmatic control of the focus object is provided;

(c) manipulation of the augmented object hierarchy is provided.

Recommended features include:

(i) support for expanding native object attributes and methods;

(ii) capability to add new interactive behaviour;

(iii) modular device installation / integration layer.

4.4.3 Abstraction

Platform abstraction refers to the capability of a platform to specify interactive behaviours by means of abstract interaction objects. Platform abstraction is necessitated due to the fact that different interaction platforms offer different programming interfaces and calling conventions, thus complicating the development of an interface that makes use of several such platforms. For platform abstraction to be possible, there should be a well defined protocol for mapping abstract interaction objects to concrete interaction elements as supported by a target platform.

Required features of a tool that supports platform abstraction include:

(a) pre-defined collection of abstract interaction object classes;

(b) for each abstract object class, a pre-defined mapping scheme to various alternative physical object classes;

(c) alternative physical instances for each abstract object class;

(d) activation of more than one physical instance for a particular abstract object class.

Recommended features of a tool that supports platform abstraction include:

(i) facilities to define new abstract interaction object classes;

(ii) methods to define alternative schemes for mapping abstract object classes to physical object classes;

(iii) methods to define run-time relationships between an abstract instance and its various concurrent physical instances;

(iv) methods to enable direct programming access, through the abstract object instance, to all associated physical instances.

4.4.5 Orthogonality

Orthogonality refers to the ability of a platform's run-time libraries to expose and make use of information produced by external software tools. When a user interacts with a particular application, there are issues that relate to the specific contexts of use and which can only be determined during the interactive session. In such cases, the interaction platform should provide the means to expose and receive information relevant to the context of use. Typically, what is required is extensions in the Application Programming Interface (API) of the interaction platform. Recommended feature is to provide support for inter-operability with other (external) software components.

5. Discussion

In this section, we discuss the results presented thus far, in the context of existing process-oriented standards in the area of software and in particular Human Computer Interaction (see section 2). The aim is to briefly review the design activities introduced in the Draft ISO 13407 on Human Centred Design (ISO, 1997b) and discuss the implications of the present work on the model presented there-within. Our objective is to link explicitly the principles of *design*

for all with Human Centred Design and to identify areas in which *design for all* extends and improves human-centred design practices.

Human Centred Design assumes four activities, which can be briefly summarised as *understanding and specifying the context of use*, *specifying user and organisational requirements*, *producing design solutions* and *evaluating designs against requirements* (ISO, 1997b). The interdependence between the four activities is depicted in Figure 7.

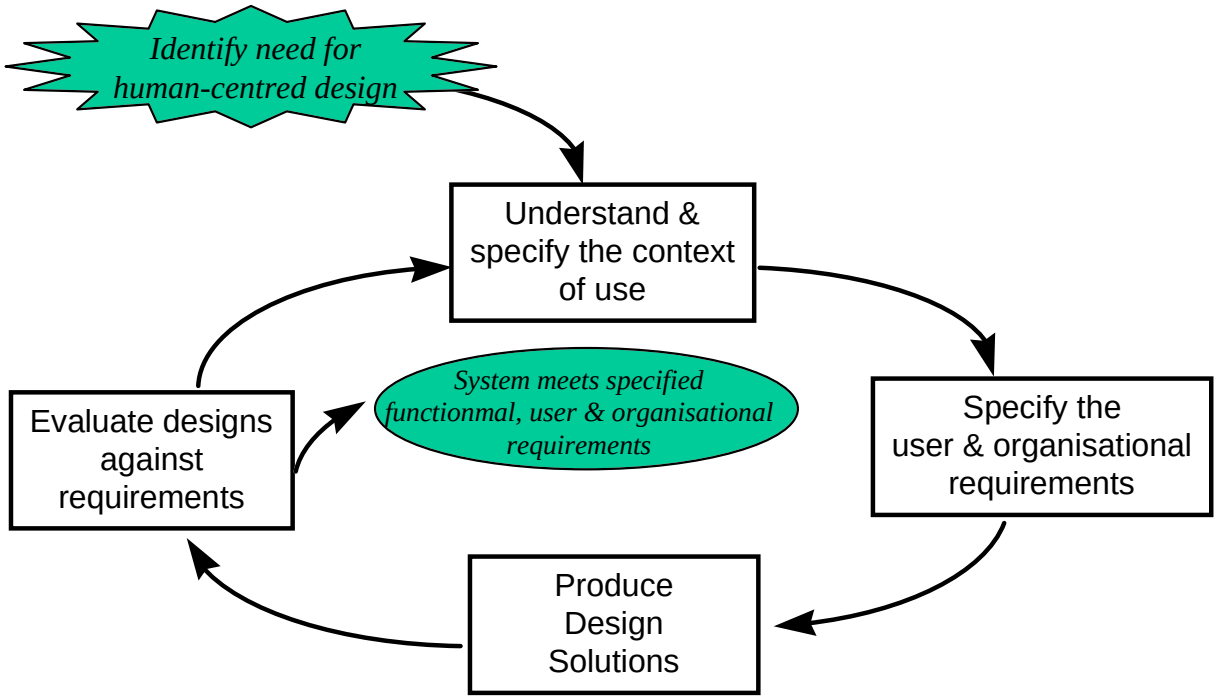

Figure 7: Interdependence of Human Centred Design activities

Though there is no doubt as to whether, or not, designing for accessibility requires a user-centred protocol such as the above, there are several remarks that need to be made with regards to the conduct of design. First of all, enumeration requires that the design outcome is not constrained to a single artefact, but rather to design spaces, containing alternative options intended for different user groups, or contexts of use. To facilitate this, the study of requirements should be broad to convey the global task execution contexts. As a result, during the first two activities, designers should elicit requirements for all target user groups using methods that are best suited for each case. There may be cases in which users do not possess a clear understanding of requirements, while in other cases, users may not be able to provide the required input. In both cases, prototyping may be part of the context and requirement elicitation inquiries to provide specifications that are subsequently translated into high fidelity prototypes.

Secondly, abstraction and rationalisation requires that design spaces are organised in such a way that they embody the required contextual information to differentiate design alternatives and specify when a particular options should be preferred from another. This body of knowledge, which is referred to as *rationale*, should clearly indicate why a design option is relevant, how does it relate to specific objectives and the circumstances, or conditions under which it should be instantiated. To this end, the objectives of evaluation should only be to test

designs against requirements, but also to assess alternatives against criteria and to provide the empirical evidence that is needed for rationalisation.

From the above, it follows that, not only the proposed guidelines are compatible with the phases of Human Centred design, but they also enrich and extend the scope of design activities to also address the rationale behind design outcomes. It is also evident that, due to the above, the protocols for assessing adherence and compliance with ISO 13407 require revisions in several directions, before the can be used to assess accessibility. In particular, they should provide additional information on the construction of the design space as well as the way in which selection is made within the respective design space. Such extensions, however, can easily be accommodated in the existing documentation intended to access conformance.

6. Concluding remarks and future work

In this paper, we have attempted to define terms and develop process-oriented guidelines and development tool requirements related to the accessibility and universal design of user interface software. The objective of the exercise is to extend the accumulated wisdom as documented in existing guidelines reference manuals on accessibility, and thus, contribute towards offering guidance on accessibility and universal design in HCI.

Specifically, this paper has presented: (a) three design guidelines that characterise how designers should proceed when confronted with the challenge of designing user interface software accessible by the broadest possible end user population, including people with disabilities; and (b) four requirements that need to be observed when making a choice of an appropriate development tool. The design guidelines have recently been embedded into the unified design method [Savidis et al, 1997], a methodological framework developed to facilitate the construction of unified interfaces [Stephanidis, Savidis and Akoumianakis, 1997]. The tool requirements discussed are also supported by the unified interface development platform [Stephanidis, Savidis and Akoumianakis, 1997] which provides the software development environment for building universally accessible user interfaces.

Acknowledgements

The material presented in this paper is based on work carried out in the context of collaborative research and development projects partially funded by the European Commission (DG XIII). These projects are:

- The TIDE-ACCESS (TP 1001) project (Development Platform for Unified Access to Enabling Environments). The partners of the ACCESS consortium are: CNR-IROE (Italy) - Prime contractor; ICS-FORTH (Greece); University of Hertforshire (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 ACTS-AVANTI AC042 project (Adaptable and Adaptive Interaction in Multimedia Telecommunications Applications). 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 Sienna (Italy), TECO Systems (Italy); STUDIO ADR (Italy); MA Systems and Control (United Kingdom).

- The TAP D4105 project (WAI - Web Accessibility Initiative). The partners of the WAI consortium are: INRIA/W3C (France) - Prime Contractor; FORTH-ICS (Greece); CNR-IROE (Italy); BrailleNet (France); EBU (France); RNIB (United Kingdom).

References

Akoumianakis, D., Stephanidis, C., 1998. *Propagating experience-based accessibility guidelines to user interface development*. Submitted for publication.

Bergman, E., Johnson, E., 1995 *Towards Accessible Human-Computer Interaction.* In Nielsen, J., (Ed.), "Advances in Human-Computer Interaction", New Jersey, Ablex Publishing Corporation, vol. 5.

Bevan, N., 1997. *Quality in Use: Incorporating Human Factors into the Software Engineering Life-cycle*. In the Proceedings of the 3rd IEEE International Software Engineering Standards Symposium and Forum, Walnut Creek, CA: IEEE Computer Society, pp. 169-179.

Casali, S. P., 1995. *A physical skills-based strategy for choosing an appropriate interface method.* In Edwards, A. D. N., (Ed.), Extra-ordinary Human Computer Interaction: Interfaces for people with disabilities, Cambridge University Press, pp. 315-342.

ISO, International Standard 9001 (1987). *Quality Systems - Model for Quality Assurance in Design, Development, Production, Installation and Servicing.* International Standards Organisation, Geneva, Switzerland.

ISO, Draft International Standard (DIS) 8402 (1994). *Quality Vocabulary*. International Standards Organisation, Geneva, Switzerland.

ISO, Draft International Standard (DIS) 9241-11 (1997a). *Ergonomic Requirements for office work with visual display terminals, Part 11: Guidance on Usability*, International Standards Organisation, Geneva, Switzerland.

ISO, Draft International Standard (DIS) 13407. (1997b). *Human-Centred Design Processes for Interactive Systems*. International Standards Organisation, Geneva, Switzerland.

Garvin, D., 1984. *What Does "Product Quality" Really Mean?* Sloane Management Review, Fall, pp. 25-48.

Norman, A., D., Draper, W., S. (Eds.), 1986. *User-centred system design: New perspectives in Human Computer Interaction*, Hillsdale, NJ: LEA.

Savidis, A., Paramythis, A., Akoumianakis, D., Stephanidis, C., 1997. *Designing user-adapted interfaces: the unified design method for transformable interactions*. In the Proceedings of the ACM Conference on Designing Interactive Systems: Processes, Methods and Techniques (DIS'97), Amsterdam, The Netherlands, 18-20 August, New York, ACM Press, pp. 323-334.

Stephanidis, C., Savidis, A., Akoumianakis, D., 1997. *Unified Interface Development: Tools for Constructing Accessible and Usable User Interfaces*, Tutorial Notes 13, 7th International Conference on Human-Computer Interaction (HCI International '97), San Francisco, California, USA, August. Available electronically at: http://www.ics.forth.gr/proj/at-hci/html/tutorials.html.

Stephanidis, C., Salvendy, G., Akoumianakis, D., 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, C., Vanderheiden, G., Weber, G., Ziegler, J., 1988. *Towards an Information Society for All: An International R&D Agenda*. International Journal of Human-Computer Interaction, vol. 10(2), pp. 107-134.

Story, M. F., 1998. *Maximising Usability: The Principles of Universal Design.* Assistive Technology, vol. 10 (1), pp. 4-12.