DRAFT INTERNATIONAL STANDARD ISO/DIS 9241-171

ISO/TC **159**/SC **4** Secretariat: **BSI**

Voting begins on: **2006-06-22**

Voting terminates on: **2006-11-22**

Ergonomics of human-system interaction —

Part 171:
Guidance on software accessibility

Ergonomie de l'interaction homme-système —

Partie 171: Lignes directrices relatives à l'accessibilité aux logiciels

(Revision of ISO/TS 16071:2003)

ICS 13.180

ISO/CEN PARALLEL ENQUIRY

The CEN Secretary-General has advised the ISO Secretary-General that this ISO/DIS covers a subject of interest to European standardization. **In accordance with the ISO-lead mode of collaboration as defined in the Vienna Agreement, consultation on this ISO/DIS has the same effect for CEN members as would a CEN enquiry on a draft European Standard.** Should this draft be accepted, a final draft, established on the basis of comments received, will be submitted to a parallel two-month FDIS vote in ISO and formal vote in CEN.

To expedite distribution, this document is circulated as received from the committee secretariat. ISO Central Secretariat work of editing and text composition will be undertaken at publication stage.

Pour accélérer la distribution, le présent document est distribué tel qu'il est parvenu du secrétariat du comité. Le travail de rédaction et de composition de texte sera effectué au Secrétariat central de l'ISO au stade de publication.

Contents

Page

Figures

Tables

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 9241-400 was prepared by Technical Committee ISO/TC 159, *Ergonomics*, Subcommittee SC 4, and by Technical Committee CEN/TC 122, *Ergonomics* in collaboration.

This first edition of ISO 9241-171 cancels and replaces, the ISO/TS 16071:2003, of which has been technically revised. During the development of ISO 9241-171, the content of ISO/TS 16071:2003 was comprehensively reviewed and revised to reflect current technologies and practices.

ISO 9241 consists of the following parts, under the general title Ergonomic requirements for office work with visual display terminals (VDTs):

- Part 1: General introduction
- Part 2: Guidance on task requirements
- Part 3: Visual display requirements
- Part 4: Keyboard requirements
- Part 5: Workstation layout and postural requirements
- Part 6: Guidance on the work environment
- Part 7: Requirements for display with reflections
- Part 8: Requirements for displayed colours
- Part 9: Requirements for non-keyboard input devices
- Part 11: Guidance on usability
- Part 12: Presentation of information
- Part 13: User guidance
- Part 14: Menu dialogues
- Part 15: Command dialogues
- Part 16: Direct manipulation dialogues
- Part 17: Form filling dialogues

ISO 9241 also consists of the following parts, under the general title Ergonomics of human-system interaction:

— Part 20: Accessibility guidelines for information communication equipment and services
— Part 110: Dialogue principles
— Part 151: Guidance on World Wide Web software user interfaces
— Part 171: Guidance on software accessibility
— Part 300: Introduction to requirements and measurement techniques for electronic visual displays
— Part 302: Terminology for electronic visual displays
— Part 303: Requirements for electronic visual displays
— Part 304: User performance test methods for electronic visual displays
— Part 305: Optical laboratory test methods for electronic visual displays
— Part 306: Field assessment methods for electronic visual displays
— Part 307: Analysis and compliance test methods for electronic visual displays
— Part 400: Principles for physical input devices; Introduction and requirements
— Part 410: Design criteria for products for physical input devices

The following parts, under the general title Ergonomics of human-system interaction, are under preparation:

— Part 411: Laboratory test and evaluation methods for the design of physical input devices
— Part 420: Selection procedures for physical input devices
— Part 421: Workplace test and evaluation methods for physical input devices

Introduction

The purpose of this part of ISO 9241 is to provide guidance on the design of the software of interactive systems to achieve as high a level of accessibility as possible. Designing human-system interactions to increase accessibility promotes increased effectiveness, efficiency, and satisfaction for people who have a wide variety of capabilities and preferences. Accessibility is therefore strongly related to the concept of usability (see ISO 9241-11).

The most important approaches to increase the accessibility of a human-system interface are:

— adopting a human centred approach to design (see ISO 13407),

— following a context-based design process,

— providing the capacity for individualisation (see ISO 9241-110),

— offering individualised user instruction and training.

It is important to incorporate accessibility goals and features into the design as early as possible as this is relatively inexpensive compared to the cost of modifying products to make them accessible once they have been designed. In addition, this part of ISO 9241 addresses the increasing need to consider social and legislative demands to ensure accessibility by removing barriers that prevent people from participating in life activities including the use of environments, services, products and information.

The guidance in this part of ISO 9241 applies to software that forms part of interactive systems used in the home, in leisure activities, in public situations, and at work. Requirements and recommendations are provided for system design, appearance and behaviour. This part of ISO 9241 provides additional guidance specifically addressing accessibility issues that supplements the software ergonomics design guidance provided by other standards, such at ISO 9241-11 to -17, and -110, and ISO 14915-1 to -3. Following the guidance in these standards is also important if the goal of accessibility is to be achieved. While the requirements and recommendations of this part of ISO 9241 are generally applicable to all software application domains, detailed guidance on the accessibility of web sites is available elsewhere [10]. These Web Content Accessibility Guidelines are consistent with the guidance provided in this Part of ISO 9241 and include some ergonomics guidance that is covered by the other ISO software ergonomics standards. Thus while the guidance in this part of ISO 9241 is applicable to the design of web sites, it covers a much wider range of applications involving the design of the software of interactive systems.

This part of ISO 9241 is based on the current understanding of the characteristics of individuals who have particular physical, sensory and/or cognitive impairments. However, accessibility is an issue that affects many people. The intended users of interactive systems are consumers or professionals with roles such as home occupiers, school pupils, engineers, clerks, salespersons, web designers etc. The individuals in such target groups vary significantly as regards physical, sensory and cognitive abilities and each target group will include people with different abilities. Thus people with disabilities do not form a specific group that can be separated out and then disregarded. The differences in capabilities may arise from a variety of factors that serve to limit the capability to engage in the activities of daily living, and are a ‘universal human experience’ [49]. Therefore, accessibility addresses a widely defined group of users including:

— people with physical, sensory and cognitive impairments present at birth or acquired during life,

— elderly people who can benefit from new products and services but experience reduced physical, sensory and cognitive capacities,

— people with temporary disabilities, such as a person with a broken arm or someone who has forgotten his/her glasses, and

— people who experience difficulties in particular situations, such as a person who works in a noisy environment or has both hands occupied by other activities.

When designing and evaluating interactive systems there are other terms that are often associated with accessibility. In Europe the term ‘Design for All’, and in North America the term ‘Universal Design’, address the goal of enabling maximum access to the maximum number and diversity of users, irrespective of their skill level, language, culture, environment or disability. This does not mean that every product will be usable by every consumer. There will always be a minority of people with severe or multiple disabilities who will need adaptations or specialised products. The term accessibility as defined in this standard emphasises the goals of maximising the number of users and striving to increase the level of usability that these users experience.

This part of ISO 9241 recognizes that some users of software will need assistive devices in order to use a system. In the concept of designing software to be accessible, this includes the capability of a system to provide connections to and enable successful integration with assistive technologies, in order to increase the number of people who will be able to use the interactive system. Guidance is provided on designing software that integrates as effectively as possible with common assistive technologies. It is important to note that accessibility may be provided by a combination of both software and hardware controlled by software. Assistive technologies typically provide specialized input and output capabilities not provided by the system. Software examples include on-screen keyboards that replace physical keyboards, screen-magnification software that allows users to view their screens at various levels of magnification, and screen-reading software that allows blind users to navigate through applications, determine the state of controls, and read text via text-to-speech conversion. Hardware examples include head-mounted pointers instead of mice and Braille output devices instead of a video display. There are many other examples not listed here. When users employ add-on assistive software and/or hardware, usability is enhanced to the extent that systems and applications integrate with those technologies. For this reason, operating systems must provide programming services to allow software to operate effectively with add-on assistive software and hardware as recommended in this standard. If systems do not provide support for assistive technologies, the probability increases that users will encounter problems with compatibility, performance and usability.

This part of ISO 9241 serves the following types of users:

— designers of user-interface development tools and style guides to be used by interface designers;

— user-interface designers, who will apply the guidance during the development process;

— developers, who will apply the guidance during design and implementation of system functionality;

— those responsible for implementing solutions to meet end user needs;

— buyers, who will reference this part of ISO 9241 during product procurement;

— evaluators, who are responsible for ensuring that products meet the recommendations of this part of ISO 9241.

The ultimate beneficiary of this part of ISO 9241 will be the end user of the software. Although it is unlikely that the end-users will read this part of ISO 9241, its application by designers, developers, buyers and evaluators should provide user interfaces that are more accessible. This standard concerns the development of software for user interfaces. However, those involved in designing the hardware aspects of user interfaces may also find the standard useful when considering the interactions between software and hardware aspects.

ISO 9241 was originally developed as a seventeen part standard on the ergonomics requirements for office work with visual display terminals. As part of the standards review process, a major restructuring of ISO 9241 was agreed to broaden its scope, to incorporate other relevant standards and to make it more usable. The title of the revised ISO 9241 – 'Ergonomics of human-system interaction' – reflects these changes and aligns the standard with the overall title and scope of SC4. The revised multipart standard is structured as a series of standards numbered in ‘hundreds’ for example the 100 series deals with software interfaces, the 200 series with human centred design, the 300 series with visual displays, the 400 series with physical input devices and so on.

Ergonomics of human-system interaction —

Part 171:
Guidance on software accessibility

1 Scope

This part of ISO 9241 provides requirements and recommendations for the design of accessible software for use at work, in the home, in education and in public places. It covers issues associated with designing accessible software for people with the widest range of physical, sensory and cognitive abilities, including those who are temporarily disabled, and the elderly. This part of ISO 9241 addresses software considerations for accessibility that complement general design for usability covered by ISO 9241-110, ISO 9241-11 to -17, ISO 14915-1 to -3 and ISO 13407.

This part of ISO 9241 is applicable to the accessibility of interactive systems. It addresses a wide range of software (e.g. office, web, learning support and library systems).

This part of ISO 9241 promotes increased usability of systems for a wider range of users. While it does not cover the behaviour or requirements for assistive technologies (including assistive software), it addresses the use of assistive technologies as an integrated component of interactive systems.

This part of ISO 9241 is intended for use by those responsible for the specification, design, development, evaluation and procurement of software operating systems and software applications.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 9241-11:1998, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability*

ISO 9241-12:1998, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 12: Presentation of information*

ISO 9241-13:1998, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 13: User guidance*

ISO 9241-14, *Ergonomics requirements for office work with visual display terminals (VDTs) – Part 14: Menu dialogues*

ISO 9241-15, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 15: Command dialogues*

ISO 9241-16:1999, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 16: Direct-manipulation dialogues*

ISO 9241-17, *Ergonomic requirements for office work with visual display terminals (VDTs) – Part 17: Form filling dialogues*

ISO 9241-110:2006, *Ergonomics of human system interaction – Part 110: Dialogue principles*

ISO 13407:1999, *Human-centred design processes for interactive systems*

ISO 14915-1, *Software ergonomics for multimedia user interfaces – Part 1: Design principles and framework*

ISO 14915-2, *Software ergonomics for multimedia user interfaces – Part 2: Multimedia navigation and control*

ISO 14915-3, *Software ergonomics for multimedia user interface – Part 3: Media selection and combination*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
accelerator keys
shortcut keys, hot keys, alternate keys, keys, key sequences, or key combinations which invoke an action immediately without displaying intermediate information (such as menus) or requiring pointer movement or any other user activity

3.2
accessibility
usability of a product, service, environment or facility by people with the widest range of capabilities

NOTE 1 The concept of accessibility addresses the full range of user capabilities and is not limited to users who are formally recognised as having a disability.

NOTE 2 The usability-orientated concept of accessibility aims to achieve levels of effectiveness, efficiency and satisfaction that are as high as possible considering the specified context of use, while paying particular attention to the full range of capabilities within the user population.

3.3
activation
initiation of an action associated with a selected user interface element

3.4
assistive technologies
hardware or software that is added to or incorporated within a system that increases accessibility for an individual

EXAMPLE Braille displays, screen readers, screen magnification software and eye tracking devices are assistive technologies.

3.5
chorded key-press
keyboard or pointer-button presses where more than one button is held down simultaneously to invoke an action

NOTE This includes both uses of modifier keys with other (non-modifier) keys as well as use of multiple non-modifier keys to enter data or invoke an action.

3.6
closed system
system that does not allow user installation of assistive technology software, due to physical or policy constraints

3.7
contrast
⟨perceptual sense⟩ assessment of the difference in appearance of two or more parts of a field seen simultaneously or successively

EXAMPLE Brightness contrast, lightness contrast, colour contrast, etc.

NOTE Adapted from CIE 17.4 definition 845-02-47

[ISO 9241-302:2006]

3.8
colour scheme
set of colour assignments used for rendering user interface elements

NOTE Colour refers to a combination of hue, saturation, and brightness.

3.9
cursor
visual indication of where the user interaction via keyboard (or keyboard emulator) will occur

NOTE Contrast with focus cursor (3.10) and contrast with pointer (3.25).

3.10
focus cursor
indicator showing which user interface element has focus

NOTE 1 Also called “location cursor”. See also input focus (3.14) and cursor (3.9).

NOTE 2 The appearance of this indicator usually depends on the kind of user interface element that has focus. The user interface element with focus can be activated if it is a control (e.g. button, menu item) or selected if it is a selectable user interface element (e.g. icon, list item).

EXAMPLE A box or highlighted area around a text field, button, list or menu options can serve as a focus cursor.

3.11
icon
graphic displayed on the screen of a visual display that represents a function of the computer system

[ISO/IEC 11581-1:2000, 4.7]

3.12
implicit designator mnemonic menu mnemonic
portion of an option name or control label used for a keyboard selection

EXAMPLE On a screen used for initiating a print job, the control label is displayed as “Print”, and the underlined “P” indicates that “P” is the implicit designator for the “Print” command.

3.13
individualisation
modification of interaction and presentation of information to suit individual capabilities and needs of users

3.14
input focus
current assignment of the input from an input device to a user interface element

EXAMPLE Pointer input focus and keyboard input focus are input foci.

3.15
keyboard emulator
software or hardware that generates input that is identical to that which comes from a keyboard.

NOTE A keyboard emulator may provide a representation of keys (e.g. on-screen keyboard) or it may not (e.g. voice recognition).

EXAMPLE Operating system-based on-screen keyboards, speech input, and handwriting are all examples of keyboard emulators if their output appears to applications as keystroke input.

3.16
keyboard focus
current assignment of the input from the keyboard or equivalent to a user interface element

NOTE For an individual user interface element focus is indicated by a focus cursor.

3.17
keyboard equivalents
keys or key combinations that provide access to functions usually activated by a pointing device, voice input, or other input or control mechanisms.

3.18
label
short, descriptive title for a user interface element.

NOTE 1 Labels include, but are not limited to, headings, prompts for entry fields, text or graphics that accompany and identify controls (e.g. displayed on the face of buttons) and audible prompts used by interactive voice response systems.

NOTE 2 In this section label refers to the presented title for a user interface element. It contrasts with the Name attribute which may or may not be presented to users but is available to assistive technologies.

EXAMPLE 1

Figure 1 — Text field with label

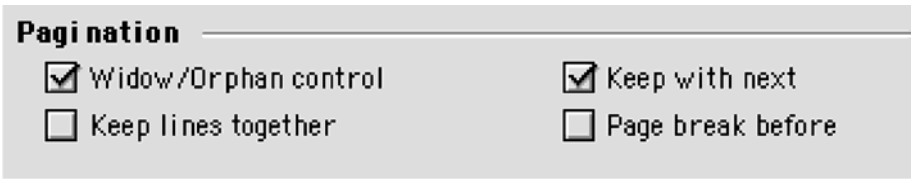

Figure 2 — Check box group, with a label for the group and a label for each check box

EXAMPLE 2 A window displays the image of a printer that the user can click to print the current document. This image's label is the image, its name attribute is "Print", its role attribute is "Push Button", and its description attribute is "A printer".

NOTE 3 In this section "label" refers to the presented title for a user interface element. It contrasts with "name" which may or may not be presented to users, but is available to assistive technologies. Textual labels are often a visual display of the name.

3.19
latch
mode in which any modifier key remains logically pressed (active) in combination with a single subsequent non-modifier keypress or pointing device button action.

NOTE Contrast with lock (3.20).

3.20
lock
persistent mode in which one or more modifier keys or pointing device buttons remains logically pressed (active) until the mode for the key or button is turned off.

NOTE 1 Unlike “latch” which affects only keyboard and pointing device actions, “lock” would affect any software that uses the modifier key(s) to alter its behaviour.

NOTE 2 Lock mode is usually turned off explicitly by the user - but may also be turned off at other times such as system shutdown or restart.

NOTE 3 Contrast with latch (3.19).

3.21
modifier key
keyboard key that changes the action or effect of another key or a pointing device

EXAMPLE 1 The shift key extends the current selection in the direction of pointer movement, rather than moving the position of the cursor.

EXAMPLE 2 Pressing “C” results in the input of that character, and pressing “Ctrl+C” results in a “Copy” function.

3.22
name
word or phrase that is associated with a user interface element and is used to identify the element to the user

NOTE 1 Names are most useful when they are the primary word or phrase by which the on screen instructions, software documentation, and its users refer to the element, and do not contain the type or status of the user interface element.

NOTE 2 When a textual label is provided it would generally present the name or a shortened version of the name. Not all user interface elements have labels however. In those cases the names would be available to assistive technologies (or sometimes by pop up tool tips, etc.).

NOTE 3 Names should not be confused with an internal identifier (ID) which may be used by software and which may not be designed to be understood by a human.

NOTE 4 See also label (3.18)

3.23
natural language
language which is or was in active use in a community of people, and the rules of which are mainly deduced from the usage

3.24
platform software
software that interacts with hardware or provides services for other software

EXAMPLE An operating system, device drivers, windowing systems, and software toolkits.

3.25
pointer
graphical symbol that is moved on the screen according to operations with a pointing device

[ISO 9241-16:1999, 3.15]

3.26
pointing device
hardware and associated software used to position the pointer

NOTE 1 Pointing devices usually have buttons that are used to activate or manipulate user interface elements.

NOTE 2 Almost any hardware can be used to control the pointer with appropriate software.

3.27
screen reader
assistive technology that allows users to operate software without needing to view the visual display

NOTE 1 Output of screen readers is typically text-to-speech or Braille.

NOTE 2 Screen readers rely on the availability of information from the operating system and applications.

3.28
usability
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

[ISO 9241-11:1998, 3.1]

3.29
user interface
all components of an interactive system (software or hardware) that provide information and controls for the user to accomplish specific tasks with the interactive system.

[ISO 9241-110:2006, 3.9]

3.30
user interface element
entity of the user interface that is presented to the user by the software.

NOTE 1 User interface elements may or may not be interactive.

NOTE 2 Both entities relevant to the task and entities of the user interface are regarded as user interface elements. Different user interface element types are text, graphics and controls. A user interface element may be a visual representation or an interaction mechanism for a task object (such as a letter, a sales order, electronic parts, or a wiring diagram) or a system object (such as a printer, hard disk, or network connection). It may be possible for the user to directly manipulate some of these user interface elements.

NOTE 3 Such elements may be containers that serve to group one or more sub-elements.

NOTE 4 User interface elements in a graphical user interface include such things as basic objects (such as window title bars, menu items, push buttons, image maps, and editable text fields) or containers (such as windows, grouping boxes, menu bars, menus, groups of mutually-exclusive option buttons, and compound images that are made up of several smaller images). User interface elements in an audio user interface include such things as menus, menu items, messages, and action prompts.

NOTE 5 UI elements are also referred to as “user interface objects”

4 Rationale and benefits of implementing accessibility

Accessibility is an important consideration in the design of products, systems, environments and facilities because it affects the range of people who are able to use them and use them easily. An accessible design is usable by a wider range of people

Accessibility can be improved by incorporating features and attributes known to benefit users with special requirements. To determine the achieved level of accessibility, it is necessary to measure the performance (effectiveness and efficiency) and satisfaction of users working with a product or interacting with an environment. Measurement of accessibility is particularly important in view of the complexity of the interactions with the user, the goals, the task characteristics and the other elements of the context of use. A product, system, environment or facility can have significantly different levels of accessibility when used in different contexts.

Planning for accessibility as an integral part of the design and development process involves the systematic identification of requirements for accessibility, including accessibility measurements and verification criteria within the context of use. These provide design targets that may be the basis for verification of the resulting design.

The approach adopted in this part of ISO 9241 has the following benefits.

The framework can be used to identify the aspects of accessibility and the components of the context of use to be taken into account when specifying, designing or evaluating the accessibility of a product.

The performance and satisfaction of the users can be used to measure the extent to which a product, system, environment or facility is accessible in a specific context.

Measures of the performance and satisfaction of the users can provide a basis for determining and comparing the accessibility of products having different technical characteristics, which are used in the same context.

The accessibility planned for a product can be defined, documented and verified (e.g. as part of a quality plan).

5 Designing for Accessibility

There are different ways of designing accessible software. This part of ISO 9241 does not assume any one specific design method or process, nor does it cover all the different activities necessary to ensure accessible system design. It is complementary to existing design methods and provides a human-centred accessibility perspective, based on ISO 13407, that can be applied, whatever the specific design process, or the particular context of use, to increase the numbers of people who are able to use the software. The guidance provided in this document is applicable at any stage in the development of an interactive system.

The development of solutions that will result in accessible software should be based upon the application of ergonomic design guidance, provided in ISO 9241-12, -13, -14, -15, -16, -17 and -110, and ISO 14915-1, -2 and -3. The guidance in these standards is also included in the provisions of other standards and guidelines (WCAG 2.0 and DFA) that specifically aim to increase accessibility. Issues that represent good ergonomics practice that are particularly important as the basis for achieving accessibility are that;

— information should be perceivable by the user (ISO 9241-12, WCAG 2.0 Principle No 1);

— content and control should be understandable (ISO 9241-12, -110 and WCAG 2.0 Principle No 3);

— interface elements should be operable (ISO 9241-110 and WCAG 2.0 Principle No 2);

— software should be error tolerant (ISO 9241-110 and DFA);

— software should be flexible in use, enabling users to choose from a wider range of input and output alternatives. (ISO 9241-110 and DFA)

The additional guidance provided in this standard specifically addresses issues that arise when providing design solutions that will satisfy the needs of users with a great variety of capabilities.

Design for accessibility should follow the principles of:

— Suitability for the widest range of use.

  Suitability for the widest range of use involves designing with the objective of producing solutions that will be useful, acceptable and available to the widest range of users within the intended user population, taking account of their special abilities, variations in their capabilities, the diversity of their tasks, and their differing environmental, economic and social circumstances.

— Equitable use

  Achieving equitable use will ensure that solutions designed to increase accessibility do not result in loss of privacy, increased risks to personal safety or security, or the stigmatisation of individuals, and that solutions provide the same means of use for all users that are identical whenever possible; equivalent when not.

— Robustness (WCAG 2.0 4th principle)

  Although it is not feasible to make all software accessible without add-on assistive technologies, these guidelines should help designers develop software that increases accessibility without the use of assistive technologies, and, by providing the necessary interface information, enables assistive software and devices to operate effectively and efficiently when used. The software can promote integration of assistive technologies by providing information that can be read by assistive technologies, and by communicating through standard application-to-application communication protocols. For example, systems that provide built-in screen magnification may enable many more users to read the text and see the images that are presented. However, if the necessary integration information is available, users may attach the screen magnification program of their choice to suit their particular needs.

6 Sources of variation in user characteristics

All user populations vary significantly in terms of their characteristics, capabilities and preferences. Any interactive system will include, within the user group for which it is designed, people with very different physical, sensory and cognitive abilities. These differences will have many different sources including innate characteristics, culture, experience and learning, as well as changes that occur throughout life. While the requirements and recommendations in this standard are based on the current understanding of the individual characteristics of people who have particular physical, sensory and cognitive impairments, their application addresses the diversity of abilities within any intended user population that may lead to limitation on activities.

Disabilities considered include not only those due to restrictions on mobility or physical performance, such as loss of a limb or tremor, but also those associated with sensory impairment, such as low vision or hearing loss, as well as cognitive factors, such as declining short term memory or dyslexia. Some disabilities may be experienced intermittently and appear unpredictably or very suddenly. In addition a disability may be experienced as an entirely uniquely event for an individual (e.g. having a broken arm). Annex C provides an outline of some of the limitations typically encountered by individuals with various types of disability, but does not constitute an exhaustive account of all the issues that may arise. In addition, limitations on activities may be created by the physical environment (e.g. poor lighting and noise), social environment (e.g. confidential tasks carried out in the presence of other people), or by the need to deal with other tasks in parallel~~,~~ and these are also taken into account.

The extent to which the particular source of any disability creates limitations varies and some of the guidance provided is specific to the degree of disability experienced. Thus impairments in vision may range from declining ability to resolve small detail to having been blind from birth. Different provisions in the design of the

interactive system may be needed to deal with different degrees of disability. For example the facility to enlarge the size of detail presented on a screen will not address the problems of those people who are blind.

It is also important to recognise that people may experience multiple disabilities. Guidance that is appropriate to addressing a specific type of disability may not work if somebody who has that disability also has some other type of disability. For example, auditory output of written text will not provide support for the deaf-blind. Combinations of different disabilities and variations in the levels of disabilities experienced can have unexpected effects. It is therefore important that different approaches to access be supported so that interfaces can be individualised to the user and their task.

7 How to use this standard

7.1 General

In order to achieve accessibility it is necessary to provide support in different parts of the software system, which includes platform software (the operating system and associated layers, and toolkits) and applications (which run on and make use of services provided by platform software). While much can be done to improve accessibility in the design of an application, it is not possible to provide all of the input and output support that users require in every circumstance at the application level alone. To the extent that any particular part of the software is dependent upon a level below it for its operational characteristics it will be necessary to ensure that the lower levels enable the implementation of recommended accessibility characteristics in any layers that depend upon them. This dependency may occur in relation to different layers in the operating system and in relation to the applications that are mounted on the operating system. Applications themselves may have layers that result in dependencies arising within different levels of the application. The majority of the requirements and recommendations in clauses 8, 9, 10 and 11 require that the issue be addressed at more than one level of the software system if the particular requirement or recommendation is to be satisfied.

In cases where the guidance is related specifically to either the operating system or application layers this is indicated in the text of the requirement or recommendation. The operating system is the general means by which accessibility features that involve control of hardware devices, in particular those involving input, are implemented and controlled. Some issues that arise in relation to the nature of the content and the form of its presentation may only apply within the application layers.

7.2 Conformance

Conformance with this standard is achieved by satisfying all the applicable requirements and by the provision of a systematic list of all the recommendations that have been satisfied. Any requirements that have been determined not to be applicable shall also be listed, together with a statement of the reasons why they are not applicable. For reference purposes all the clauses including requirements are listed in Informative Annex A.

Users of this standard should consider each requirement (a “shall” statement) and recommendation (a “should” statement) to determine whether it is applicable in the particular context of use that has been established for the interactive system that is being designed. ISO 13407:1999, 7.2 and ISO 9241-11:1998, 5.3 provide guidance on the identification and specification of the context of use. Guidance on the determination of applicability is provided in Informative Annex B.

Informative Annex B provides a means both for determining and recording the applicability of all the requirements and recommendations and means for reporting that they have been followed. Other equivalent forms of report are acceptable.

Server software (used in client-server and mainframe environments) shall be evaluated in conjunction with the client (including terminal) software that would be used with it.

Software used on or intended to be used on Closed Systems shall be evaluated in conjunction with the intended hardware configuration and conform to the requirements in all clauses, except 8.6.

8 General guidelines

8.1 Input/output alternatives

8.1.1 Enable user input/output choice

Platform software should enable input and output alternatives in different modalities.

EXAMPLE 1 Software allows users to read output as graphics, text, Braille or speech, and allows users to provide input by a keyboard, mouse, tablet, voice or Braille device.

EXAMPLE 2 An end user with a haptic or tactile disability can understand the tactilely presented message because the message is also available in electronic text.

EXAMPLE 3 A user with work-related disorders in the shoulder or upper extremity may need to switch between input modes in order to alleviate pain.

8.1.2 Enable switching of input/output alternatives

Platform software should enable users to switch among input/output alternatives without requiring them to reconfigure or restart the system or applications, unless there would be no change of state or data.

NOTE This capability aids users with different abilities who are working together on the same system.

EXAMPLE 1 A person who is blind uses their system only through the keyboard, using keyboard substitutes for mouse actions. A sighted user working on the same system can use the mouse and type in text. The system does not have to be restarted in between each session.

EXAMPLE 2 While working on the same system without making permanent configuration changes, a user points at an user interface element and speaks the word “Print” to print the contents, while another user presses a key sequence, and a third user chooses a menu item.

8.2 Names and labels for user interface elements

8.2.1 Provide a name for each user interface element

Software shall associate an identifying name with every user interface element except where the name would be redundant.

NOTE 1 A name conveys the identity of the user interface element to the user. It complements the role attribute that tells an element’s function (such as that it acts as a push button) and the description attribute that summarizes the element’s visual appearance.

NOTE 2 Names would be redundant for user interface elements whose entire informational content is already conveyed by their role attribute (such as a horizontal rule), static text elements that serve to name other elements, and elements that only serve as an integral portion of a parent element (such as the rectangular border around a button).

NOTE 3 In some cases the name will be displayed visibly, but in other cases it will only be provided programmatically for use by assistive technology as described in 8.6.

NOTE 4 Such elements may be containers that serve to group one or more sub-elements. In a typical graphical user interface, examples of UI elements include basic elements such as window title bars, menu items, push buttons, images, text labels and editable text fields, while examples of containers include windows, list boxes, grouping boxes, menu bars, menus, groups of mutually-exclusive option buttons, and compound images that are made up of several smaller images. In a typical audio user interface, examples of interactive UI elements include menus, menu items, messages, prompt tones, and pauses.

EXAMPLE 1 The application provides a label showing the name "File name" for a static text field that shows the name of the file being described in the fields below.

EXAMPLE 2 The software does not need to provide a name for a static text field that says "Last name:" and serves to identify text box that follows it, as that string would be exposed using the field's value attribute.

EXAMPLE 3 The software does not need to provide a name for an image that forms part of a raised border around a non-standard button.

EXAMPLE 4 Compound user interface elements that consist of a collection of other user interface elements have a group name. A Web page image composed of a series of smaller image files provides a group name for a "Construction Site") in addition to the names of the individual component images ("building", "bulldozer", "dump truck", "crane", etc.).

EXAMPLE 5 A control is listed in the product documentation as "the Print button," and its identifying name in the software is "Print" (regardless of whether the word "Print" appears on the visual representation of the button or not).

EXAMPLE 6 Dialog boxes or windows have meaningful names, so that a user who is hearing rather than seeing the screen gets appropriate contextual information.

8.2.2 Provide meaningful names

Names of user interface elements should be comprised of natural language words that are meaningful to the intended users.

NOTE 1 This means that each word in the name would occur in a standard dictionary or in electronic documentation for end-users included with the software.

NOTE 2 The names are most useful when they are the primary name by which the software, its documentation, and its users refer to the element and do not contain the type or status of the user interface element.

NOTE 3 User interface elements that represent a real entity (such as a documents, location, or person) may be provided with the name of that entity even if the name is too lengthy or cryptic to be read easily.

NOTE 4 Names may use terms that are specific to a particular task domain provided that they have established meanings for the intended users.

EXAMPLE 1 A user interface control is listed in the product documentation as "the Print button," and its identifying name in the software is "Print" (regardless of whether the word "Print" appears on the visual representation of the button or not).

EXAMPLE 2 Dialog boxes or windows have meaningful names, so that a user who is hearing rather than seeing the screen gets appropriate contextual information.

8.2.3 Names provided by the system should be unique

Each name of a user interface element specified by the software developer should be unique within its context.

NOTE 1 Users will not be able to use the name to identify an element if several elements have the same name within the same context.

NOTE 2 A name is considered unique if no other user interface element with the same name and role attributes shares the same container or parent element (such as a window, group box, section, etc.)

NOTE 3 User interface elements that represent a real entity (such as a documents, location, or person) may be provided with the name of that entity even if the name is too lengthy or cryptic to be read easily.

EXAMPLE 1 A dialog box has three text fields where the user can enter file names, and each box has an associated Browse button. Since all three buttons have the same label ("Browse"), a group box is put around each pair of associated controls, and the group boxes are given labels such as "File 1", "File 2", and "File 3".

EXAMPLE 2 A form for a purchasing application has several rows of items, each containing a text field displaying the title of a book, followed by "Buy" button that is used to purchase that book. Even though the face of each button looks identical, the form is implemented so that each provides a unique name for use by assistive technology, such as "Buy The Grapes of Wrath" and "Buy Pride and Prejudice".

EXAMPLE 3 An application has several rows of items, each containing a text field displaying the title of a book, followed by "Buy" button that is used to purchase that book. Even though the labels of all the buttons are identical, the form is implemented so that assistive technology can programmatically determine the relationship between a button and the text field that serves as its visual label.

EXAMPLE 4 A word processing application displays two windows showing the same document. In order to distinguish between them, the number "2" is appended to the title of the second window. Each window displays a dialog box with the title "Document Properties," and even though the two windows have the same title, each forms a unique pair when used with the titles of their parent windows.

8.2.4 Make names available to assistive technology

Each name of a user interface element and its association shall be made available by the software system to assistive technology in a documented and stable fashion.

EXAMPLE In a platform that does not provide a service for the association of names and elements, an application documents how assistive technologies can access that information.

8.2.5 Display names

If a user interface element has a visual representation and is not part of standard components of the user interface, software should present its name to end users (either by default or at the user's request).

EXAMPLE 1 A button with the name “Print” displays the word “Print” on its surface

EXAMPLE 2 A print button has a picture of a printer with a text label that pops-up when the user pauses a pointer over the button and also when the user moves focus to the button and presses a specific keyboard command

8.2.6 Provide short names and labels

Each name or label of a user interface element specified by the software developer should be brief enough to be rendered audibly without inconveniencing the user.

NOTE 1 Developers are encouraged to put the most distinctive parts of the name first, so users can skip over the latter parts once they have read enough to identify the element.

NOTE 2 User interface elements that represent a real entity (such as a documents, location, or person) may be provided with the name of that entity even if the name is too lengthy or cryptic to be read easily.

EXAMPLE 1 “Print” is used instead of “Print button” or “This button prints the current document.”

EXAMPLE 2 An icon representing a document is labelled with the file name or title of the document, even if that string is too lengthy or cryptic to be read easily, because that string was determined by the document author rather than by the software developer.

8.2.7 Provide text label display option for icons

Every icon provided by software should have an associated text label and the user should be given the option of choosing between displaying only the icon image, the icon image with the text label or only the icon text label.

NOTE It is useful for the user to be able to adjust the font size (see also 10.3.1).

8.2.8 Properly position the labels of user interface elements on screen

The labels for user interface elements provided by software should be consistently positioned, relative to the elements that they are labelling, on the display. (ISO 9241-12:1998, 5.9.4 and 5.9.6).

If the platform software has conventions for positioning labels these conventions should be followed, relative to the elements that they are labelling.

NOTE 1 This helps assistive technology correctly associate the labels with their corresponding controls, and helps users of screen enlargers know where to look for a label or control.

8.3 User preference settings

8.3.1 Enable easy individualisation of user preference settings

Software should enable user preference settings to be easily individualised to meet users' needs and tasks.

NOTE 1 The ability to individualise is frequently important for achieving accessibility.

EXAMPLE 1 A software application allows users to configure and save settings for font size and style within a particular window.

NOTE 2 Requiring users to hand edit a configuration file is not an easy method for individualising preference settings because it is too easy for the user to accidentally enter invalid values or otherwise corrupt the file.

NOTE 3 In many cases software can support system-wide user preference settings rather than implementing software-specific user preference settings.

EXAMPLE 2 A user chooses preference settings through a graphical user interface, rather than directly editing the configuration files.

8.3.2 Enable adjustment of sizes and colours of common user interface elements

Software should enable users to adjust the size and colour of common user interface elements including, but not limited to, window-title font sizes, window border colours, window border thickness, and window controls, if applicable to the task.

NOTE Platform software often supports these options for standard user interface elements it provides.

8.3.3 Enable individualisation of the user interface look and feel

Software should provide a mechanism enabling users to individualise the interface look and feel including the modification or hiding of command buttons.

EXAMPLE 1 A user with a cognitive disability may, when using a given application, change the interface via a "skin" to simplify the application's look and feel.

EXAMPLE 2 A word processor allows users to hide menu items and tool bar buttons that they do not find useful.

8.3.4 Enable individualisation of the cursor and pointer

If the hardware supports the service, software shall enable users to individualise attributes of all cursors and pointers including but not limited to shape, size, stroke width, colour, blink rate (if any), and pointer trails (if any).

NOTE 1 The ability to set the cursor to non-blinking is important for users with attention deficits, who may be easily distracted.

NOTE 2 The colour aspect of this provision is not applicable if the presentation of the cursor or pointer is an inversion of the image and it has no colour.

NOTE 3 Platform software often supports these options for the standard cursors and pointers it provides.

EXAMPLE 1 Users with low vision can change a cursor from non-blinking to blinking, and adjust the size to be more readily visible given their visual capabilities.

EXAMPLE 2 Users with low vision and/or a colour deficiency can change the thickness and colour of the focus cursor so that they can more easily see the current input focus.

EXAMPLE 3 Users with low vision can make the pointer larger so that they can more readily locate it.

8.3.5 Provide user-preference profiles

Software should enable users to create, save, edit and recall profiles of preferences, including input and output characteristics, without having to edit configuration files or carry out any restart that would cause a change of state or data.

NOTE 1 For systems that provide access for multiple users, such as library systems, conversion back to a default profile may be advisable.

NOTE 2 It is often useful to be able to access the preferences over a network. Doing this in a secure way would preserve privacy especially for people with disabilities who are worried about identification.

NOTE 3 It is advisable to minimize the need to restart the system or application in order for changes in user interface settings to become effective.

EXAMPLE 1 Platform software allows each user to save global settings for font size, sound volume, and pointer-control settings that apply everywhere on the system.

EXAMPLE 2 A software application allows users to configure and save settings for font size and style within a particular window.

EXAMPLE 3 The profile for a public library system is modified for the needs of a current user but returns to default values when that user is finished.

EXAMPLE 4 A person is completing an online process and has to make adjustments to the accessibility feature to reduce errors. Restarting the operating system or the user agent would cause them to lose their work.

8.3.6 Provide capability to use preference settings across locations

Software should permit users to transfer their preference settings easily onto a compatible system.

NOTE 1 Portability is important for users with disabilities because they may find a system difficult or impossible to use without the preferences set to meet their interaction needs. The overhead and effort required to create preference settings can be a significant hindrance to system usability if it must be repeated at every location.

NOTE 2 User preferences profiles are sometimes made publicly available e.g. for download from internet. Because people can be concerned about others knowing of their disability, it would be helpful if their use of these resources could be kept private.

NOTE 3 Some platform software may provide a general mechanism for transferring their preference settings; in such cases software may not have to implement this feature itself, as long as it follows platform conventions for storing its user preference settings.

EXAMPLE 1 A user visiting a different building on the company network logs in and the system automatically locates and uses his or her personal preference settings from the network without having to edit configuration files.

EXAMPLE 2 A user loads a preference settings file from a floppy disk onto a new computer.

EXAMPLE 3 A users preference settings are loaded from a smart card onto a new system.

8.3.7 Enable user control of timed responses

Unless limits placed on the timing of user responses are essential to maintaining the integrity of the task or activity or are based on real life time constraints (e.g. an auction), software shall allow users to adjust each software-specified user response time parameter in one or more of the following ways:

— the user is allowed to deactivate the time-out, or;

— the user is allowed to adjust the time-out over a wide range which is at least ten times the length of the default setting, or;

— the user is warned before time expires, allowed to extend the time-out with a simple action (for example, "hit any key") and given at least 20 s to respond.

EXAMPLE A logon prompt requires the user to enter their password within 30 s. Show the remaining time on the screen and provide a control to stop the time decrementing.

8.4 Special considerations for accessibility adjustments

8.4.1 Make controls for accessibility features discoverable and operable

Software shall enable the On/Off controls and adjustments for accessibility features to be discoverable, and operable by those who need that feature.

NOTE Features are discoverable if their settings and description can be found by browsing the user interface (including browsing Help available through the application).

EXAMPLE 1 Rather than needing to use a combination of keys, a user can turn the “StickyKeys” accessibility feature on and off by pressing the Shift key five times in succession.

EXAMPLE 2 Accessibility features are turned on by use of a single toggle key held down at system start-up.

EXAMPLE 3 Controls for individualising low-vision options are shown in large type by default.

8.4.2 Safeguard against inadvertent activation or deactivation of accessibility features

Software should prevent inadvertent activation or deactivation of accessibility features.

EXAMPLE The software system requests confirmation before activating or deactivating accessibility features.

8.4.3 Avoid interference with accessibility features

Software shall not disable or interfere with the accessibility features of the operating system and software that provide services for use by other software or any other applications.

EXAMPLE 1 Software which intercepts keyboard input does not defeat the operation of keyboard filters such as key latching and locking.

8.4.4 Inform user of accessibility feature On/Off status

Software should make the current status of accessibility features available at all times.

EXAMPLE 1 A control panel shows the current state of all accessibility features.

EXAMPLE 2 A small icon on the screen indicates that an accessibility feature is switched on.

8.4.5 Inform user of accessibility feature activation

Software should inform users when activating an accessibility feature that would interfere with standard operation in default configuration, and provide an opportunity to accept or cancel the activation.

8.4.6 Enable persistent display

When users can activate a menu, control, or other user-interface element to display additional information or controls, software shall allow that information or control to persist while the user engages in other tasks, until the user chooses to dismiss it [39], if it is appropriate to the task.

NOTE Persistent display of frequently used windows and controls may be helpful for users who have physical, language, learning or cognitive disabilities, and reduces the number of steps required to access them.

EXAMPLE 1 Users can keep a Help Window available as they go through the tasks described.

EXAMPLE 2 The user can “tear off” one or more menus and continue to view and/or use them while navigating and using other menus.

EXAMPLE 3 The user can add a toolbar button that duplicates the function of a specific menu command.

8.5 General control and operation guidelines

8.5.1 Optimise the number of steps required for any task

Software should be designed to optimise the number of steps that the user has to perform for any given task

NOTE It is important to find a balance between reducing the steps to improve efficiency and adding steps to allow for sufficient explanation of an infrequent task. ISO 13407 provides guidance on human centred design processes that can help find the optimal design.

EXAMPLE 1 A user who wants to print one document can do so in only two steps. Once they select the Print icon on the toolbar, a dialogue is displayed (which can, if desired, be used to change various print settings) and the user simply selects the OK button.

EXAMPLE 2 A user with cerebral palsy types very slowly, and so finds it much more convenient when they can save a document by pressing a single key combination rather than having to navigate menus and dialog boxes.

8.5.2 Provide “Undo” and/or “Confirm” functionality

Software should provide a mechanism that enables users to undo at least the most recent user action and/or cancel the action during a confirmation step [39].

NOTE 1 Although this is a general ergonomic principle, “Undo” mechanisms are particularly important for users who have disabilities that significantly increase the likelihood of an unintentional action. These users can require significant time and effort to recover from such unintentional actions.

NOTE 2 A macro is considered to be one user action.

NOTE 3 Generally, the more consecutive actions the user can undo, the better.

NOTE 4 It is preferable if undo operations themselves can be undone.

NOTE 5 This may not be possible for such interactions as operations which cause fundamental transformation of logical or physical devices, or may involve a data exchange with 3rd parties that are out of the software’s control, etc.

NOTE 6 It is preferable that the default configuration provides a confirmation step for any actions that the user cannot undo with a single Undo command. Software may allow the user to disable the confirmation for specific actions.

EXAMPLE 1 A user with Parkinson’s disease may inadvertently input a sequence of keystrokes, which activate several dialogues that need to be undone. The use of several steps of the undo function may permit the user to go back to the original state.

EXAMPLE 2 A user is about to format a hard disk. As this is an operation that cannot be undone, the software shows a confirmation dialog before the formatting begins.

8.5.3 Provide accessible system start-up and restart

If a task requires user interaction when the state of the software prevents use of assistive technology or speech output (such as during system start up), the software shall provide the user with an alternative means, which does not require user interaction during that period, for completing the task.

NOTE System start-up and restart are fundamental functions of the system and they should be accessible. This includes all operations prior to the stage where the user's accessibility aids and preference settings are available.

EXAMPLE 1 A computer is configured with a small "boot loader" that lets the user to choose between two or more operating systems present on the system. Because the boot loader's menu is run before a full operating system or any assistive technology is running, it provides a mechanism by which the user can, during a normal session and using their assistive technology, specify which operating system will be loaded the next time the system starts.

EXAMPLE 2 A computer does not request any password or other user input until after it has loaded access features.

EXAMPLE 3 A public information kiosk restarts automatically and comes up accessible. There is no log on before access features work.

8.5.4 Enable software-controlled media extraction

If the hardware allows for it, software shall enable the user to perform software-controlled media extraction.

NOTE 1 Those media include, but are not limited to, floppy disks, CD-ROMs and DVDs.

NOTE 2 In most cases the user can use this feature as provided by the platform software without explicit support from software applications.

EXAMPLE A user who cannot press physical buttons on the computer or handle a CD-ROM can nevertheless use their on-screen keyboard to instruct the operating system to eject the disc, so that it will not interfere with the computer's next restart.

8.5.5 Support “Copy” and “Paste” operations

Software should support “Copy” and “Paste” operations for all user interface elements that support text input.

NOTE 1 “Copy” and “Paste” operations enable users with disabilities to avoid awkward, slow and error-prone manual re-entry of potentially large amounts of data.

NOTE 2 Copy operations from password entry fields and similar secure objects can copy the text that is displayed on the screen rather than the actual text.

EXAMPLE As the user types into a password entry field, the field displays a dot to represent each character typed. The user can select this text and copy it to the clipboard, but the appropriate number of asterisk characters are copied rather than the typed password.

8.5.6 Support “Copy” operations in non-editable text

Software should support “Copy” operations for all user interface elements that display text.

NOTE The ability to copy non-editable text into the clip board can help users with disabilities avoid awkward, slow, or error-prone manual input of that text into other locations.

EXAMPLE A user who communicates with the user assistance team by email provides examples of problems by copying the text from an error dialog box and pasting it into the email rather than having to retype it.

8.5.7 Enable selection of elements as an alternative to typing

Where the user can enter commands, file names, or similar choices from a limited set of options, software should provide at least one method for selecting or choosing that does not require the user to type the entire name.

NOTE 1 This reduces cognitive load for all users, and reduces the amount of typing for users for whom spelling or typing is difficult, slow, or painful.

NOTE 2 In many cases these features are supported automatically when software incorporates standard user interface elements provided by the platform software.

EXAMPLE 1 An application prompts the user for a filename by presenting a dialog box in which the user can type in a filename or choose from a list of existing files

EXAMPLE 2 At a command line prompt, the user can type the first letter or letters of a file name and then press TAB to complete the name. Repeatedly pressing TAB would cycle through the names of additional files that match the string the user entered. This same mechanism can be used to enter command names as well as file names.

8.5.8 Allow warning or error information to persist

Software shall ensure that error or warning information shall persist or repeat in a suitable manner as long as the source of the error or warning remains active, or until the user dismisses it ([39], [44] and [46]).

EXAMPLE A dialog box indicating that a file was not saved remains visible until the user presses a “Close” button.

8.5.9 Present user notification using consistent presentation techniques

Alerts, warnings, and other user notification should be presented by software using consistent presentation techniques that enable a user to locate them and identify the category of the information (e.g., alerts versus error messages).

EXAMPLE 1 A “beep” and positioning messages are provided in a consistent place that allows users who have low vision to look for and find the error message more easily.

EXAMPLE 2 Every error message occurs in a dialog box, while every informative (non-error) message appears in the bottom left of a window. The consistent position of error messages allows users who are viewing only part of the screen through magnification to predict where particular types of information are likely to be found.

8.5.10 Provide understandable user notifications

Alerts, warnings, and other user notifications provided by software should be short, simple and written in a clear language.

NOTE 1 Short messages do not preclude the provision of additional details on request.

NOTE 2 ISO 9241-13:1998 provides detailed recommendations on user guidance.

EXAMPLE 1 Notifications are presented in the language of the user, avoiding internal system codes, abbreviations, and developer-oriented terminology.

EXAMPLE 2 A message box appears, displaying the short, meaningful message "The network has become unavailable." The user can choose the OK button to dismiss the message, or choose the "Details" button to see the more detailed message "Error #527: thread 0xA725 has failed to yield mutex during maximum timeout period" which may or may not be comprehensible to the user.

8.6 Compatibility with assistive technology

8.6.1 General

The provisions in this section are intended to provide the information and programmatic access needed by assistive technologies to help users access and use software. These provisions would only apply to systems that allow installation of assistive technology or where assistive technology will be installed in conjunction with the software.

8.6.2 Enable communication between software and assistive technology

Platform software shall provide a set of services that enable assistive technologies to interact with other software sufficient to enable compliance with guidelines 8.6.5, 8.6.6, 8.6.7, 8.6.8, and 8.6.9 [27] and [37]).

If accessibility services are provided by the operating system, software toolkits shall make these services available to their client software.

NOTE 1 Assistive technologies can use these accessibility services to access, identify, or manipulate the user interface elements of an application. Applications can use these services to provide information about its user interface elements and automation facilities to other software.

NOTE 2 In some cases it is possible for the assistive technology to be running on a remote system.

EXAMPLE 1 A screen reader uses an accessibility service to query information about a non standard user interface element which is displayed at a particular location.

EXAMPLE 2 A screen magnifier uses an accessibility service to receive notifications of focus changes in applications so it can always display the user interface element which has the focus.

EXAMPLE 3 Speech recognition software uses the accessibility services to first get information about the custom toolbar of an application and then activate one of the elements of that toolbar.

8.6.3 Use standard accessibility services

Software that provides user interface elements shall use the accessibility services provided by the system to cooperate with assistive technologies. If it is not possible to comply with 8.6.6, 8.6.6, 8.6.7, 8.6.8 and 8.6.9, using these means, the software shall use other services that are supported, and publicly documented, and implemented by assistive technology.

NOTE 1 In many cases the standard user interface elements provided by platform software already make use of the accessibility services, so the application only has to explicitly use the accessibility services when it uses non-standard user interface elements.

NOTE 2 Assistive technology may be running on the same system or on a separate system from the application software.

EXAMPLE 1 An application having non-standard user interface elements uses the accessibility services of the operating system to provide information about the name, description, role, state, etc of those user interface elements.

EXAMPLE 2 A word processing application uses the accessibility services to provide access to the text of the document being edited. For instance it can inform about the cursor position, the character, word or sentence at the cursor position, the content of the current selection, etc.

EXAMPLE 3 An application uses the accessibility services to send notifications when its user interface changes, so that assistive technologies can update their internal representation of the screen state.

EXAMPLE 4 A company is developing a productivity application for a platform that does not provide any standardized method for letting applications communicate with assistive technology. The company determines that there is no toolkit available that would supply this functionality. They contact developers of assistive technology programs for that platform, and in cooperation with them design, implement, and publish a communication mechanism that each product implements.

8.6.4 Make user interface element information available to assistive technologies

Software shall provide assistive technology with information about individual user interface elements, using methods compatible with 8.6.3, except elements that only serve as an integral portion of a larger element, taking no input and conveying no information of their own.

NOTE 1 User interface element information includes, but is not limited to: general states (such as existence, selection, focus, and position), attributes (such as size, colour, role and name), values (such as the text in a static or editable text field), states specific to particular classes of user interface elements (such as On/Off, depressed/released), and relationships between user interface elements (such as when one user interface element contains, names, describes, or affects another). This applies to on screen user interface elements and UI status values such as toggle keys.

NOTE 2 User interface element information is typically available to users by inspection or interaction. Users with certain disabilities may not be able to see or otherwise detect this information without using assistive technologies.

NOTE 3 In many cases these features are supported automatically when software incorporates standard user interface elements provided by the platform software.

NOTE 4 See 8.2, "Names and Labels for User Interface Elements" for more information on the name property and the relationship between an element and its visual label, element name/label properties.

NOTE 5 See 8.6.7 for how an application uses the accessibility services to send notifications when its user interface changes, so that assistive technologies can update their internal representation of the screen state.

EXAMPLE 1 A person with dyslexia can have the text on the screen read to them, and highlighted as it is read, because a screen reader utility can determine the text along with word, sentence, and paragraph boundaries.

EXAMPLE 2 A blind user presses a keyboard command asking their screen reader to tell them where they are working. The screen reader uses accessibility services to ask the current application for the identify of the user interface element that has the keyboard focus, then queries that element parent or container, and repeats it all the way to the main application window. It then generates artificial speech saying "Down option button, Direction group box, Find dialog box, Status Report dot text dash Notepad application."

EXAMPLE 3: A blind user can have the text on the screen voiced aloud to them by a screen reader utility, which uses a separate voice to indicate when the font, font size, or colour changes. It also uses that voice to tell them when it reaches an embedded picture, and reads the picture's description if the author provided one.

EXAMPLE 4: A blind user can also ask their screen reader to voice the word and character at the current insertion point, and the text that is currently selected.

EXAMPLE 5 A user presses a keyboard command asking their macro utility to move the keyboard focus upwards on the screen. The utility asks the current application for the focus element and its location, and then checks other locations above that point until it finds a user interface element that can take the focus. It then programmatically sets the focus to that element.

EXAMPLE 6 Speech recognition software uses the accessibility services to identify the application's toolbar and the controls on it, and adds the names of those controls to its active vocabulary list. When it hears the user say "Click Save" it activates the toolbar's Save button. (It was able to determine that the control's name was "Save" even though it visually appears as a picture of a floppy disk.)

EXAMPLE 7 Developers build an application using standard controls provided by the operating system. Because those controls already include support for the platform's accessibility services, the developers only need to make sure they provide names and a few other attributes for those controls. They make sure their pre-release testing includes users who rely on assistive technology, who verify that the application works with their products.

EXAMPLE 8 An application having non-standard user interface elements uses the accessibility services of the operating system to provide information about the name, description, role, state, etc of those user interface elements.

EXAMPLE 9 Software provides assistive technology with information about a scroll bar, including its type (Scroll Bar), name (Vertical), value (47%), size and location. This allows screen reader software to describe it to the user. The application also provides information about the individual components of the scroll bar that can be independently manipulated, including the Up button, Down button, Page Up button, Page Down button, and Position Indicator. This allows users to click, drag, and otherwise manipulate those components using speech recognition programs.

EXAMPLE 10 Software does not bother to provide assistive technology with information about individual lines that are used to visual represent a control. Assistive technology can determine the control's boundaries by querying its size and location attributes, so it does not need to rely on the position of individual drawing elements.

EXAMPLE 11 When displaying tabular data or data in columns, the application provides assistive technology with information about the data, including any row or column names.

8.6.5 Allow assistive technology to change focus and selection

Software shall allow assistive technology to modify focus and selection attributes of user interface elements, using methods as specified in 8.6.3.

NOTE In many cases these features are supported automatically when software incorporates standard user interface elements provided by the platform software.

EXAMPLE Speech recognition software listens for the user to speak the name of a user interface element in the current application window. Once it hears a matching name, it wants to give focus and selection to that user interface element. The application and operating system allow the speech recognition software to do this directly. It does this because it may not be clear to the speech recognition software how to move the focus to the user interface element using simulated keystrokes or mouse movements.

8.6.6 Provide user interface element descriptions

Where tasks require access to the visual or audible content of user interface elements beyond what the role and name attributes provide software shall provide descriptions of those objects, that are meaningful to the user and available to assistive technology through a standard programmatic interface (as described in 8.6.3), whether those descriptions are presented or not [44].

NOTE 1 In contrast with the label attribute that names a user interface element (as described in section 8.2), and the role attribute that identifies its function, the description should convey the visual appearance of the element, and is only needed when the label and role attributes are insufficient to allow the user to fully interact with the element.

NOTE 2 Visual user interface elements that are purely decorative and contain no information need not be described. However, elements that at first appear decorative may in fact have an information function, such as acting as a separator, an icon, a visual label, etc. In such cases the element should be provided with role and/or label attributes as described in 8.6.4.

NOTE 3 Users who have low vision or are blind may use software that can present text descriptions to users who cannot view visually displayed user interface elements.

NOTE 4 Descriptions also assist communication between people who use the visual display and people who use assistive technology.

EXAMPLE 1 Alice instructs Bob to click on the picture of a pencil. Bob is blind, but his screen reader utility program tells him that the button labeled 'Compose' has the description 'A picture of a pencil', so Bob instructs his screen reader to activate that button.

EXAMPLE 2 A map image has a label “Map of Europe”, with a description “A map depicts Western Europe, with a jagged line across France and Germany indicating where the glacial advance stopped in the last Ice Age.”

EXAMPLE 3 A graphic encyclopædia’s animation (dynamic object) provides a stored textual description: “A lava flow pours from the volcano, covering the town below it within seconds”.

EXAMPLE 4 An audio presentation of a visual display, provides a summary of the available content prior to presentation of the content details: “This page contains five images and two text paragraphs”.

8.6.7 Make event notification available to assistive technologies

Software shall provide assistive technology with notification of events relevant to user interactions, using methods described in 8.6.3.

NOTE 1 Events relevant to user interaction include, but are not limited to, changes in user interface element status (such as creation of new user interface elements, changes in selection, changes in focus and changes in position), changes in attributes (such as size, colour and name), and changes of relationships between user interface elements (such as when one user interface element contains, names, describes or affects another). Just as important are input events, such as key presses and mouse button presses, and output events, such as writing text to the screen or playing audio information. This also applies to user interface status values (such as the states of toggle keys).

NOTE 2 In many cases these features are supported automatically when software incorporates standard user interface elements provided by the platform software.

EXAMPLE 1 When a user selects an item in a list box, assistive software is notified that a selection event has occurred in the list box.

EXAMPLE 2 When a user changes the position of an icon, assistive software is notified that the icon has changed position.

EXAMPLE 3 When a user causes a push-button to gain focus, assistive software is notified that focus has changed to that button.

EXAMPLE 4 When a user changes the position of a pointer or cursor, assistive software is notified that the position has been changed.

EXAMPLE 5 When an audio is playing, notification is sent to assistive technology that generates speech so that speech output will not conflict with the audio.

EXAMPLE 6 When the Caps Lock becomes active, either in response to the user pressing a key or through a programmatic action, a screen reader can be notified and inform the user who cannot see the status light on the keyboard.

8.6.8 Allow assistive technology to access resources

If mechanism(s) exist, software should provide assistive technology with access to shared system resources.

NOTE Such resources include, but are not limited to, processor time, space on the display, control of and input from the pointer and keyboard, and system-wide shortcut keys. This is important so that the user is not prevented from effectively using assistive technology in conjunction with an application or tool.

EXAMPLE 1 Speech recognition software running in the background receives enough processor time to keep up with the user's speech because the foreground application does not try to use all available processor time. (When software does try to use all available processor time, it lets accessibility aids override or supersede that behaviour.)

EXAMPLE 2 A screen magnifier can display a window that is always visible on the screen, because applications do not insist on obscuring all other windows. (When software does obscure other windows, including docked toolbars, it lets accessibility aids override or supersede that behaviour.)

EXAMPLE 3 The user can move the pointer over the window of an on-screen keyboard utility, because the active application does not restrict the pointer to its own window.

EXAMPLE 4 The user can use a screen magnifier and a voice recognition utility at the same time because, even though both are controlled by keyboard commands even when they do not have the focus, the user can configure them so they do not rely on the same key combinations.

EXAMPLE 5 A keyboard macro utility can monitor the user's keystrokes because applications avoid using low-level functions that read input directly from the keyboard and would bypass the layers that the macro package relies on.

EXAMPLE 6 The user is able to view instructions in one window while carrying them out in another, because neither window insists on taking up the entire screen

8.6.9 Use system-standard input/output

Software shall use standard input and output methods provided by the software platform, or, if this is not possible, make equivalent information available through means described in 8.6.3.

NOTE These capabilities are also useful in enabling automated testing applications, pervasive macro/scripting facilities, and the use of intelligent agent software.

EXAMPLE 1 Software moves a cursor using system routines. This allows assistive software to read the current cursor position.

EXAMPLE 2 Software bypasses the system routines for graphic drawings for better performance. The software provides an option that detects the state of an "assistive technology flag". When the flag is set, the software uses the system routines for graphics.

8.6.10 Enable appropriate presentation of tables

When presenting information in the form of tables, or multiple rows or columns, information about layout, orientation, and relationships between the data represented shall be communicated to assistive technology using methods discussed in provision 8.6.3 ("Use standard accessibility services") [47].

EXAMPLE When presenting the table, information about the orientation and relationship between the data represented in rows and columns is communicated to a screen reader.

8.6.11 Accept the installation of keyboard and/or pointing device emulators

Software shall accept the installation of keyboard and/or pointing device emulators that work in parallel with standard input devices.

NOTE It is important that the pointing device alternatives work in parallel with the regular pointing device. A user with low motor function might move the mouse pointer to the general vicinity of a target, and then fine-tune the position using the alternate pointing device. There may also be multiple users of the machine.

EXAMPLE 1 The operating system accepts a button-based mouse emulator that can be used at the same time as the standard mouse.

EXAMPLE 2 The operating system accepts a mouse-based on-screen keyboard emulator that can be used at the same time as, or independently of, the physical keyboard.

8.6.12 Allow assistive technology to monitor output operations

Platform software shall provide a mechanism that allows assistive technology to monitor standard output operations at the level that software makes such requests, and to identify the source and result of each operation.

EXAMPLE 1 An operating system provides services by which applications draw text to the screen, but the operating system internally converts the text to an image before passing it to the display driver. The operating system therefore provides services by which a screen reader can get notified about drawing operations, and examine both the original text and the location at which they will be displayed, before they are converted to images.

EXAMPLE 2 A graphics toolkit provides services by which applications draw to and otherwise manipulate bitmap images in memory, and later copy those images to the screen. The toolkit provides services by which a screen reader can get notified about those drawing operations, so that it can keep track of the shapes, text, and pictures visible in the image.

EXAMPLE 3 Assistive technology monitors separate output to the left and right audio channels so that a training application can inform the user of important spatial information.

8.7 Closed systems

8.7.1 Read content on closed systems

Software that is on or intended for installation on closed systems shall allow the user from a keyboard or keypad to move focus to any visually presented information and have that content read aloud.

8.7.2 Announce changes on closed systems

Software that is on or intended for installation on closed systems shall allow the user to have any change in focus, status, or content audibly announced.

8.7.3 Operable through tactilely discernable controls

All functionality of the software shall be operable through the tactilely discernable controls of the hardware without requiring vision.

8.7.4 Pass through of system functions

Software that is on or intended for installation on closed systems shall pass through or implement the operating system and software that provide services for use by other software accessibility features.

9 Inputs

9.1 Alternative input options

9.1.1 Provide keyboard input from all standard input mechanisms

Platform software should provide a method for generating keyboard input from each standard input mechanism provided by the platform.

EXAMPLE 1 A platform that supports mouse input and includes a mouse-operated on-screen keyboard utility that can be used to control any application that is designed to take keyboard input.

EXAMPLE 2 A platform provides a built in speech recognition feature and the facility to type any key or key combination on the standard keyboard using the speech recognition.

9.1.2 Provide parallel keyboard control of pointer functions

Platform software shall provide a keyboard alternative to standard pointing devices that enables keyboard (or keyboard equivalent) control of pointer movement and pointer button functions in parallel with the standard pointing device ([39] and [44]).

NOTE 1 This is commonly called MouseKeys (see Annex D).

NOTE 2 This allows users who have restricted limb/hand movement or coordination to more easily control pointing functions.

NOTE 3 It is important that the keyboard alternative works in parallel with the regular pointing device (mouse, trackball, touchscreen, etc.). A user with low motor function might move the mouse pointer to the general vicinity of a target, and then fine-tune the position using the keyboard control.

9.1.3 Provide pointer control of keyboard functions

Platform software should provide a pointing device-based alternative to the keyboard that enables pointing device control of key presses latching, and locking.

NOTE This allows users who cannot use the keyboard and can only use a pointing device to type.

EXAMPLE 1 A person who cannot use the keyboard can operate the device completely with a head operated mouse.

EXAMPLE 2 An operating system includes an on-screen keyboard emulator that allows the user to perform the equivalent of pressing, latching, and locking all keyboard keys using only a pointing device.

9.1.4 Provide speech recognition services

If the hardware has the capability to support speech recognition, platform software should provide or enable the use of programming services for speech recognition.

NOTE 1 This does not imply that a speech recognition engine should always be installed.

NOTE 2 This is relevant for users with visual, physical, and cognitive disabilities.

EXAMPLE A virtual machine allows software that it hosts to access speech recognition services provided by the operating system.

9.2 Keyboard focus

9.2.1 Provide focus cursor

Software shall provide a focus cursor that visually indicates which user interface element currently has the keyboard input focus, as well as the focus location within that element when one exists.

NOTE The availability of this information to Assistive Technology is covered under assistive technology compatibility clause 8.6.6.

EXAMPLE 1 A box or highlighted area appears around the checkbox that will be activated if the user hits the space bar.

EXAMPLE 2 A cursor (a flashing I bar) appears in the data entry field at the location where any typed characters will be inserted and also at the end of a text selection highlight to show which end of the highlight will move with the next shift-arrow-key stroke.

EXAMPLE 3 A range of text is selected. The text cursor appears at the end of the selected text action.

9.2.2 Provide high visibility focus cursor

At least one focus cursor shall be visually locatable by people with unimpaired vision at 2,5 m when software is displayed on a 38 cm (15 inch) diagonal screen at 1024 x 768 pixels resolution, without moving the cursor.

9.2.3 Restore state when regaining focus

When a window regains focus, software should restore the focus, selection, and active modes to the values they had before the window lost the focus.

NOTE This is important because, if focus is not retained when a window regains focus via keyboard navigation, a keyboard user must press many keystrokes to return to their previous location and selection.

EXAMPLE 1 A user is currently focused on the third button in a window. Focus is switched to another window. When the user returns to the original window, focus should be returned to the third button in that window.

EXAMPLE 2 The user is editing the contents of a spreadsheet cell. Focus is switched to another window. When the user returns to the original window, the application is still in editing mode, the same text is selected, and the keyboard input focus is at the same end of the selected text as it was before the window lost focus.

9.3 Keyboard input

9.3.1 General

Although the guidelines in this sub clause refer to “keyboard input”, the source of such keyboard input may be a variety of software and hardware alternative input devices.

In this sub clause, the term “keyboard” should be interpreted as referencing a logical device rather than a physical keyboard

9.3.2 Enable full use via keyboard

Unless the task requires time-dependent analogue input software shall provide users with the option to carry out all tasks using only non-time dependent keyboard (or keyboard equivalent) input.

NOTE 1 This includes, but is not limited to, editing text and other document components, and navigation to and full operation of all controls.

NOTE 2 This provision is particularly critical for people unable to use pointing devices.

NOTE 3 Operating system-based on-screen keyboards, speech input, and handwriting are all examples of keyboard equivalents since their output appears to applications as keystroke input

EXAMPLE 1 A water color painting where the darkness is dependent on the time the cursor spends at any location is exempt because the task requires time-dependent analogue input.

EXAMPLE 2 Users move input focus among and between windows displayed from different software using voice commands that generate only keyboard input.

NOTE 4 Use of MouseKeys would not satisfy this guideline because it is not a keyboard equivalent to the application; it is a mouse equivalent (i.e. it looks like a mouse to the application).

NOTE 5 All input functionality needs to be keyboard operable, but not necessarily all UI elements. If there are multiple ways to perform a task, only one of them needs to be keyboard operable, though it is best if all possible forms are keyboard operable.

NOTE 6 This does not preclude and should not discourage the support of other input methods (such as a mouse) in addition to keyboard operation.

NOTE 7 This includes accessibility features built into the application.

EXAMPLE 3 Features for people who are hard of hearing are operable from the keyboard because people who have hearing disabilities may also have physical disabilities that prevent them from using a mouse.

NOTE 8 This requirement also includes keyboard navigation between groups of controls, as well as, inside those groups (described in more detail in requirement 9.3.17).

EXAMPLE 4 A user uses the tab key to navigate to and from a list, and uses the up/down arrow keys to navigate up and down within the list.

EXAMPLE 5 All user interface elements or equivalent functions accessible via the pointer are accessible via keyboard input. Users make menu choices, activate buttons, select items, and perform other pointer-activated tasks via keyboard input.

EXAMPLE 6 A computer-aided drawing (CAD) program is usually used with a mouse, but also provides the facility to specify points by x, y, and z coordinates, and to select drawing elements from a hierarchical list of sub-assemblies and parts or by using arrow keys to navigate through elements displayed on the screen. Users navigating via the keyboard have no problem identifying the position of the focus.

EXAMPLE 7 An application in a PDA is usually operated with a stylus, but all functionality can also be controlled from any keyboard that plugs into the PDA.

EXAMPLE 8 An educational physics simulator for the parabolic movement of a launched object uses the mouse to simulate the angle (direction) and strength (speed) of the object launching. This is a highly intuitive input method, but inconvenient for people unable to use the mouse. An alternative input method could be a form in which the user writes down the values of angle and strength.

9.3.3 Enable sequential entry of multiple (chorded) keystrokes

Software shall enable users to lock or latch modifier keys (e.g. Shift, Control, Alt, Option) so that multiple key combinations and key-plus-mouse button combinations can be entered sequentially rather than by simultaneously pressing multiple keys ([39], [41] and [44]).

NOTE 1 Most operating systems provide this function for all standard modifier keys.Other software is generally responsible for implementing this feature only for non-modifier keys that it treats as modifier keys.

NOTE 2 This is commonly called StickyKeys (see Annex D).

NOTE 3 This allows users who have physical impairments a means to enter combination key commands (e.g., Ctrl-C, Ctrl-Alt-Del) by pressing one key at a time.

EXAMPLE A graphics program allows the user to modify mouse clicks by holding down the “Del” key. Because “Del” is not treated as a modifier key by the operating system, the graphics program either provides key latching and locking, or provides an alternative method that does not require simultaneous operations.

9.3.4 Provide adjustment of delay before key acceptance

Software that handles low-level keyboard input shall enable users to adjust the delay during which a key is held down before a key-press is accepted across a range of times that includes a value of two seconds (2s) ([39], [41] and [44]).

NOTE 1 This is commonly called SlowKeys (see Annex D).

NOTE 2 A common range for a key acceptance delay feature is 0,5 s to 5 s.

NOTE 3 This feature allows users who have limited coordination, and who may have trouble striking an intended key, to input the intended key-press by holding the key down for a longer period of time than unintended key-presses. This delay of acceptance means that short key-presses caused by bumping keys unintentionally are ignored.

NOTE 4 BounceKeys (9.3.6) would have no effect if SlowKeys is active

9.3.5 Provide adjustment of same-key double-strike acceptance

Software that handles low-level keyboard input shall enable users to adjust the delay after a keystroke, during which an additional key-press will be ignored if it is identical to the previous keystroke across a range of times that includes a value of one-half second (0,5 s) ([41] and [44]).

NOTE 1 This is commonly called BounceKeys (see Annex D).

NOTE 2 This feature allows users, who may have tremors or other motor conditions that cause them to unintentionally strike the same key more than once, to prevent a system from accepting inadvertent key-presses.

NOTE 3 BounceKeys" would have no effect if SlowKeys is turned on.

NOTE 4 A typical range for a double strike acceptance delay feature is 0,2 s to 1 s.

9.3.6 Provide adjustment of key repeat rate

The software system should enable users to adjust the rate of key repeat.

NOTE This feature allows users with slow reaction time to better control the number of repeated characters that will be produced by holding down a key during some time.

9.3.7 Provide adjustment of key-repeat onset

Software that handles low-level keyboard input should enable users to adjust the time between the initial key press acceptance and key repeat onset across a range of times including a value of two seconds (2 s).

NOTE This prevents users whose reaction time may be slow from producing unwanted repeated characters by holding down a key long enough to unintentionally initiate key repeat.

9.3.8 Allow users to turn key repeat off

Software that handles low-level keyboard input that supports key repeat shall enable users to turn off the key repeat feature.

NOTE This feature prevents users with very slow reaction time from producing unwanted repeated characters while holding down a key.

9.3.9 Provide notification about toggle-key status

Software should provide information to the user in both visual and auditory form concerning changes to the status of keys that toggle or cycle between states [39].

NOTE 1 This is commonly called ToggleKeys (see Annex D).

NOTE 2 This allows users who are unable to see keyboard status lights to determine the current state of a binary-state keyboard toggle control such as “Caps Lock” or “Num Lock.”

NOTE 3 Applications do not need to duplicate the functionality of the ToggleKeys feature provided by major operating systems. Applications that are developed for a specific platform need to provide feedback for all keys that toggle or cycle that are used by the application and not already handled by a ToggleKeys feature provided by the platform.

NOTE 4 8.6.7 (Make event notification available to assistive technologies) requires software to notify assistive technology of these status changes.

EXAMPLE 1 A locked state is indicated by a high-frequency beep (or two tone mid - high sequence) and an unlocked state with a low-frequency beep (or a two tone mid - low sequence).

EXAMPLE 2 Firmware in a notebook computer generates three different tones as the user presses the Fn and F4 keys to cycle between three different projection states (LCD, CRT, and LCD+CRT).

EXAMPLE 3 An application uses the INSERT key to toggle between inserting typed characters and having them replace existing text. This mode is specific to the application and therefore not handled by the operating system’s ToggleKeys feature, so the application toggles an indicator in its status bar and optionally generates a tone to indicate whether insertion is On or Off

9.3.10 Provide accelerator keys

Software should provide accelerator keys for frequently used features ([39] and [44]).

NOTE 1 In many cases, not every feature can or needs to be mapped to an accelerator key. The choice of what features to map to accelerator keys may be made by determining which features would constitute a core set of frequent and useful functions.

NOTE 2 Accelerator keys are especially important for users who type slowly, interact only through a keyboard, or use keyboard emulators such as speech-recognition systems. Users who have disabilities benefit because they can reduce time-consuming steps that would otherwise be required to activate accelerated features.

NOTE 3 The decision about which keys can be used as accelerators is dependent on the conventions on the platform and the language of the user interface.

EXAMPLE User can press “Ctrl-C” to copy, “Ctrl-V” to paste or “Ctrl-P” to print.

9.3.11 Provide implicit designators

Software should provide implicit designators that are displayed by default for all user interface elements that take input and have visible textual labels.

NOTE This does not preclude providing an option to turn off the implicit designators.

EXAMPLE 1 In the portion of a menu shown in Figure 3, the implicit designators are the underlined letters: “T”, “S”, “E”, “W”, “g”, “m”, “L” and “D”.

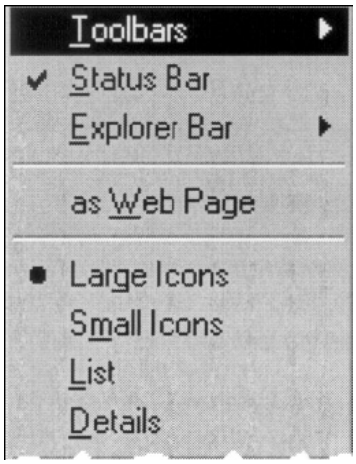

Figure 3 — Example of implicit designators in a menu

EXAMPLE 2 In the pushbuttons shown in Figure 4, the implicite designators are "D" and "P":

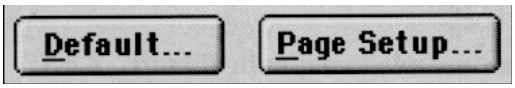

Figure 4 — Example of implicit designators in pushbuttons

EXAMPLE 3 On a screen used for initiating a print job, the control name is displayed as “Print”, and the underlined “P” indicates that “P” is the implicit designator, for the “Print” command.

9.3.12 Reserve accessibility shortcut key assignments

The shortcut key assignments in the left hand column of Table 1 shall be reserved for the purposes shown in the right hand column of Table 1:

Table 1 — Reserved shortcut key assignments

Shortcut key	Used for:
Five consecutive clicks of shift key	On/Off for StickyKeys
Shift key held down 8 seconds	On/Off for SlowKeys and RepeatKeys

NOTE To accommodate other accessibility options, platform software may reserve additional shortcut keys.

9.3.13 Enable remapping of keyboard functions

Software should enable users to re-assign the mappings of all keys.

EXAMPLE A user who has a left arm and no right arm switches frequently used letters from the right to the left side of the keyboard.

9.3.14 Separate keyboard navigation and activation

Software shall allow users to move the keyboard focus without triggering any effects other than the presentation of information (e.g. scrolling or pop-ups that do not change the focus or selection). An explicit keystroke or similar user action shall be provided to trigger any other user-initiated effect.

NOTE 1 This does not preclude the provision of additional navigation techniques that do cause effects such as changing selection.

NOTE 2 This is particularly important for users who cannot see the entire screen at one time, and so must explore the user interface by navigating through all available user interface objects. In some cases they would not be aware of any side effects caused by such navigation.

EXAMPLE 1 A user presses the Tab key to move from a button to a set of checkboxes. When the first checkbox acquires focus, it does not become activated. Activation requires a separate step, such as pressing the spacebar.

EXAMPLE 2 Using arrow keys to navigate through items in a list automatically selects the item that receives the focus, so the application allows the user to hold down the CTRL key while navigating to move the focus without changing the selection.

9.3.15 Follow operating system keyboard conventions

Software should follow keyboard access conventions established by the platform software on which it is run.

NOTE 1 This improves the usability of new applications, but it is especially relevant for people who can only use the keyboard or have cognitive impairments.

NOTE 2 The platform conventions normally include the assignment of implicit designators, modifier keys and accelerator keys.

NOTE 3 This does not preclude the provision of additional keyboard shortcuts and techniques in addition to those that are operating system conventions.

NOTE 4 Keyboard conventions may be established by the operating system or by a separate graphical user interface layer.

EXAMPLE 1 An application follows the system conventions that “Alt” is used to indicate the use of implicit designators when held down and to activate the application main menu, and pressed and released.

EXAMPLE 2 An application avoids reassigning the key combination used by the operating system to activate the MouseKeys feature.

EXAMPLE 3 An application uses the “Esc” key to cancel its custom dialog and message boxes, because that follows the convention established by the operating system.

9.3.16 Facilitate long list and menu navigation

Software should provide keyboard mechanisms to facilitate navigation within long menus, lists, and documents.

NOTE Wrapping with auditory and visual indication of rollover is one strategy. HOME and END keys are another. Often both are provided.

EXAMPLE 1 The user presses “Home” to move to the first item in a list, “End” to move to the last item in the list, and “PgUp” and “PgDn” to move forward and backward the number of items currently visible.

EXAMPLE 2 The user types one or more characters to move to the next item that starts with those characters.

9.3.17 Arrange input/output controls in task-appropriate groups

Controls should be arranged so that when the user navigates with the keyboard they are visited in appropriate order for the user's task [46].

NOTE For users who are visually impaired or blind, the order and grouping in which keyboard navigation occurs may be the only order in which they can use controls.

EXAMPLE As a user presses the tab key, the focus cursor moves to a task-appropriate group of radio buttons, followed by the next group of radio buttons, and so on, in a task and conceptually appropriate order. Within each group of radio buttons, the user moves among related buttons by pressing the arrow key

9.4 Pointing devices

9.4.1 General

The term "pointing device" in this clause refers to any physical or logical pointing device. Such devices include mice, trackballs, touchscreens, and touchpads, as well as specialized input devices such as head trackers, "sip & puff" systems, and many other hardware/software combinations that systems treat as pointing devices. Some devices, such as touchscreens and touchpads, may use a finger tap or gesture in place of physical buttons, and these should be interpreted as equivalent to pointer-button events and covered by provisions addressing such types of input.

9.4.2 Provide direct control of pointer position from external devices

Platform software shall provide a service to enable software, including pointing device drivers, to directly position the pointer. In addition, all pointing device drivers shall support direct positioning of the pointer.

EXAMPLE An eye-gaze mouse alternative plugged into USB and using the standard mouse driver can set absolute position of the mouse pointer on the screen.

9.4.3 Provide easily-selectable pointing device targets

Target size should be optimized by the software system to maintain adequate target selectability, grouping and separation from adjacent user interface elements.

NOTE This makes usage easier for all pointing device users, but it is especially important for enabling users with disabilities to select user interface elements effectively with a mouse or head operated pointing device.

9.4.4 Enable the reassignment of pointing device button functions

Platform software shall enable users to reassign the functions for each pointing device button [39].

NOTE It is desirable for applications to respect global OS settings for button re-assignments rather than making such assignments on a per application basis.

EXAMPLE 1 A user with partial paralysis in the right arm wishes to remap the position of the mouse buttons to use it with their left arm. Instead of buttons being interpreted as button 1, 2, and 3 from left to right, they can be remapped as buttons 3, 2 and 1 reading from left to right.

EXAMPLE 2 A user with a trackball that has four buttons can choose which positions to use for each function based on their ability to reach them.

9.4.5 Provide alternative input methods for complex pointing device operations

Software should enable all user initiated actions that can be accomplished with multi-clicks (i.e., double or triple clicks), or simultaneous pointing device operations (e.g. hold and drag), or spatial or temporal gestures (e.g. scribbling motion or holding buttons down for designated periods of time) to be accomplishable with an alternative pointing device method that does not require multi-clicks, simultaneous operations or gestures.

NOTE The number of buttons available on standard devices may limit the ability to use pointing device buttons as 'multi-click' or 'button-hold' buttons. Thus other methods are usually needed to accomplish multi-click or simultaneous mouse button actions.

EXAMPLE 1 User can use the right click menu to achieve the same function as the double click.

EXAMPLE 2 Instead of holding the pointer button down to keep a popup open, the user can click to open it and click again to close it.

9.4.6 Enable pointing device button-hold functionality

Software shall provide a method such that users are not required to hold down a pointing device button for more than the system single-click time in order to directly manipulate a user interface element, activate a control, or maintain a view of a menu.

NOTE 1 In many systems, this facility may have to be built into driver software.

NOTE 2 If “MouseKeys” feature is implemented fully in parallel with mouse – then this functionality would be provided since you could use the keyboard to hold down the pointer device buttons.

EXAMPLE 1 Users have the option to view a menu by pressing and releasing a mouse button rather than pressing and holding it.

EXAMPLE 2 Users have the option to “lock” single-clicks so that they are treated as continuous button presses, allowing them to select across text without holding down a mouse button.

EXAMPLE 3 Using MouseKeys (which allows the user to press a key on the number pad to lock the mouse button down), users can drag and drop user interface elements without continuously pressing down on a mouse button.

9.4.7 Provide adjustment of delay of pointer-button-press acceptance

Software that handles low-level pointing device input should enable users to adjust the delay during which a pointing device button is held down before a button-press is accepted, across a range of times including a value of one second (1 s).

EXAMPLE A user who has tremors sets a duration time sufficient to prevent tremor-induced unintentional presses from being accepted as intentional presses as they move the mouse around.

9.4.8 Provide adjustment of minimum drag distance

Software that handles low-level pointing device input should enable users to adjust the minimum pointer movement while the pointer button is held down that will be registered as a drag event.

EXAMPLE A user who has tremors is able to select an item using a mouse without accidentally dragging that item to a new location.

9.4.9 Provide adjustment of multiple-click parameters

Software that handles low-level pointing device input shall enable users to adjust the interval required between clicks, and the distance allowed between the positions of the pointer at each click, to accept the operation as a double- or triple-click [39].

EXAMPLE 1 A user with slow movements may take several seconds between clicks in a double click intended to open a document.

EXAMPLE 2 Users with tremor often inadvertently move the mouse cursor between the first and second click of a double click. Because they can adjust the distance allowed between two clicks of a double click they can choose a distance that allows them to successfully double click even with their inadvertent movement between clicks.

9.4.10 Provide adjustment of pointer speed

Software that handles low-level pointing device input shall enable users to adjust the speed or ratio at which the pointer moves in response to a movement of the pointing device [39].

EXAMPLE Users may change the speed of the pointer movement by setting an absolute speed or a ratio between movements of the pointing device and the pointer, so that the pointer movement is changed from a 1:1 mapping between the movement of the pointing device and the pointer to a 3:1 mapping.

9.4.11 Provide adjustment of pointer acceleration

If software provides pointer device acceleration it shall provide adjustment of the pointer movement acceleration including a setting of zero.

NOTE Zero acceleration level allows programmed mouse movements from alternate mouse device at high speed.

9.4.12 Provide adjustment of pointer movement direction

Software that handles low-level pointing device input should enable users to adjust the direction at which the pointer moves in response to a movement of the pointing device.

NOTE 1 The pointer movement options include, but are not limited to being the same, the opposite or perpendicular to the pointing movement direction.

NOTE 2 This is useful for people with movement limitations.

9.4.13 Provide a means of finding the pointer

Platform software shall provide a mechanism to enable users to locate the mouse pointer, unless it is always high contrast with background, always visible, and always solid and larger than text.

EXAMPLE A user with low vision looses track of the mouse pointer. When the Control key is pressed animated concentric circles are presented around the location of the mouse pointer.

9.4.14 Provide alternatives to simultaneous pointer operations

Software shall provide a non-chorded alternative for any chorded key or button presses, whether chorded presses are on the pointing device alone or are on the pointing device in combination with a keyboard key-press [39].

NOTE The intent here is to replace or supplement concurrent actions with sequential input alternatives, because multiple simultaneous actions may be difficult or impossible for users with motor impairments.

EXAMPLE 1 If a task can be performed by pressing mouse button 1 and mouse button 2 simultaneously, it also can be performed using one mouse button to display a menu providing the same function.

EXAMPLE 2 If a file can be copied by pressing a keyboard modifier key while holding down a mouse button and dragging, then it is also possible to perform this task by selecting a menu operation called “copy”.

10 Outputs

10.1 General output guidelines

10.1.1 Avoid seizure-inducing flash rates

Software shall avoid flashing that may cause photo-sensitive seizures.

NOTE Refer to international and national standards for photosensitive seizure disorders for detailed parameters.

10.1.2 Enable user control of time-sensitive presentation of information

Whenever moving, blinking, scrolling, or auto-updating information is presented, software shall enable the user to pause or stop the presentation.

NOTE 1 Individuals with low vision or reading problems need time to study information in order to comprehend it.

NOTE 2 Varying speed of presentation is also useful.

EXAMPLE 1 A user controls the tactile output by pausing its presentation to be allowed to read and to continue again.

EXAMPLE 2 A user presses mouse down on animated text, which pauses it as long as they hold the mouse down to read it.

10.1.3 Provide accessible alternatives to task relevant audio and video

When task relevant information is presented by audio or video, software shall provide equivalent content in accessible alternative formats.

EXAMPLE 1 A video includes captions for the auditory track.

EXAMPLE 2 The system provides an auditory description of the important information of the visual track of a multimedia presentation. (This is called Audio Description.)

10.2 Visual output (displays)

10.2.1 Enable users to adjust graphic attributes

To increase legibility of graphics, software should enable users to change attributes used to present the content without changing its meaning [39].

NOTE There are numerous cases where changing the view will necessarily change the meaning. The intent is that users be enabled with the capability to change views as much as possible without changing the meaning.

EXAMPLE 1 A user who has low vision wishes to view a line graph of the stock market averages over the past five years. To see the graph, the user changes the thickness and colour of the line.

EXAMPLE 2 The user can change attributes, such as line, border, bullet size and shadow thickness, for improved viewing of charts, graphs and diagrams, but such changes would not affect the meaning.

EXAMPLE 3 The length of a temperature gauge does not change unless the scale was lengthened proportionally.

EXAMPLE 4 A user changes the size of icons making it easier for them to tell them apart.

10.2.2 Provide a visual information mode usable by users with low visual acuity

Software should provide at least one mode for visual information usable for users with corrected visual acuity between 20/70 and 20/200, without relying on audio.

NOTE One possibility is that the software magnifies what is shown in the screen. Another one is to enable the user to change the size of fonts and icons.

10.2.3 Use text characters as text, not as drawing elements

In graphical interfaces, text characters should be used as text only, not to draw lines, boxes or other graphical symbols [39].

NOTE 1 Characters used in this way can confuse users of screen readers.

NOTE 2 In a character-based display or region, graphic characters may be used.

EXAMPLE A box drawn with the letter “X” around an area of text is read by screen-reader software as “X X X X X” on the first line, followed by “X” and the content and “X”. Text used for graphics in this way is usually confusing or uninterpretable when read sequentially by users with assistive software.

10.2.4 Provide keyboard access to information displayed outside the physical screen

If the virtual screen (e.g. desktop) is made larger than the visible screen, so that some information is off screen, the platform software shall provide a mechanism for accessing that information from the keyboard.

NOTE A viewing area extending the physical boundaries of the computer is usually called a virtual screen

EXAMPLE A moving view-port allows the users to pan to see the virtual screen area not displayed on the physical screen

10.3 Text/Fonts

10.3.1 Enable users to set minimum font size

Software should enable users to set a minimum font size with which information would be presented on the display (despite its font size in the document) ([4], [39] and [44]).

NOTE If the platform software already provides this facility, the application may utilize it.

EXAMPLE 1 A word processor contains a “draft mode” which shows all document text in a single, user-selectable font, colour, and font-size, overriding any formatting information specified in the document itself. When the user encounters small text that they have difficulty reading, they can switch into this mode and will still be viewing the same section of the document, but at a size they have already selected as meeting their needs.

EXAMPLE 2 A user has difficulty reading small text on the screen, so they set a “minimum font size” preference value in the operating system’s control panel. Their Web browser respects this setting and automatically enlarges any text that would otherwise be smaller than this size.

10.3.2 Adjust the scale and layout of user interface elements as font-size changes

User-interface elements should be scaled or have their layout adjusted by software as needed to account for changes in embedded or associated text size.

NOTE 1 This also applies to text associated with icons (see [4], [39] and [44]).

NOTE 2 In many cases these features are supported automatically when software incorporates standard user interface elements provided by the platform software.

EXAMPLE 1 As fonts grow, button and menu sizes adjust to accommodate them. If they become large enough, the window increases in size to prevent buttons from clipping (overwriting) each other. If the window would otherwise become too large to fit on the visible portion of the display, scroll bars are added.

EXAMPLE 2 A user increases the operating systems’ global setting for the number of screen pixels per logical inch. An application then displays a window that was designed to contain an image below three lines of 10-point text. However, because of the global setting change the 10-point text is now drawn using a larger number of physical pixels, and so is taller than in the default configuration. The application takes care to measure the height of the text when deciding where to draw the icon, rather than assuming the text will be a predictable number of pixels tall.

10.4 Colour

10.4.1 Do not convey information by colour output alone

Software shall not use colour alone as the only way to convey information or indicate an action.

NOTE See ISO 9241-12:1998, 7.5.1.

EXAMPLE 1 Red is used to alert an operator that the system is inoperative or indicate an emergency situation. In these cases, the use of colour is supplemented by text indicating “warning” or “emergency.”

EXAMPLE 2 If an indicator changes colour to show an error condition, then the user can also get text or audio information that indicates the error condition.

EXAMPLE 3 Negative numbers are coded in red and also have parentheses.

10.4.2 Provide colour schemes designed for people who have visual impairments

Software that includes colour schemes should provide colour schemes designed for use by people who have visual impairments [46].

EXAMPLE High-contrast monochrome schemes are provided, including one using light foreground on dark background and another using dark foreground on light background. The software system also includes schemes that avoid the use of colours that may confuse users who have common forms of colour blindness, cataracts, macular degeneration and other visual impairments.

10.4.3 Provide individualisation of colour schemes

Software that uses colour schemes should allow users to create, save and individualise colour schemes, including background and foreground colour combinations.

NOTE 1 Ability to share schemes is also useful.

NOTE 2 See 8.3 for provisions dealing with the individualisation and persistence of these settings

EXAMPLE 1 A user adjusts the colour scheme provided for those with red-green colour blindness to optimise discriminability for their particular requirements.

EXAMPLE 2 A person with low visual acuity uses the operating system’s control panel to request that window captions and menus be drawn in yellow text on a black background.

EXAMPLE 3 The user can choose the colors scheme that IS used to draw different types of user interface elements (such as windows, menus, alerts, cursors, and default window background and text), indicators for general states (such as focus and selection), and codings for task-specific states (such as online, offline, or error)

10.4.4 Allow users to individualise colour coding

Except in cases where warnings or alerts have been standardised for mission-critical systems (e.g. red = network failure), software should allow users to individualise colours used to indicate the selection, process, and the types, states, and status of user interface elements.

EXAMPLE 1 If a user chooses red as the colour to represent links, an embedded application should not override that setting and use another colour.

EXAMPLE 2 A user who cannot discriminate between red and green can set the printer-status colours to be dark blue for OK and yellow to indicate printer problems. In addition the system provides an auditory warning if there is a problem with the printer.

10.4.5 Provide contrast between foreground and background

Default combinations of foreground and background colours (hue and luminance) of the software should be chosen to provide contrast regardless of colour perception abilities.

EXAMPLE Colours are selected for contrast differences so that they are distinguishable, on the basis of light/dark differences, by users who cannot discriminate between different hues.

10.5 Window appearance and behaviour

10.5.1 Provide meaningful window titles

Every window should have a meaningful title not shared with any other window currently displayed by the same software, even if several windows display multiple views of the same user interface element.

EXAMPLE A user views a document in a window titled "Draft" and then opens a second view of that same document. The second window is titled "Draft: 2".

10.5.2 Provide unique window titles

Platform software should ensure that all windows have titles not shared with any other window currently on the system.

EXAMPLE When software creates a new window or changes the title of an existing window, the operating system checks the name of all other windows on the system and, if there is a conflict, appends a unique number to the title.

10.5.3 Enable non-pointer navigation to windows

Software shall enable users to use the keyboard or other non-pointer input mechanisms to move focus to any window currently running that is allowed to accept focus.

NOTE The intent here is to allow users who cannot use a pointing device to navigate among windows with a keyboard in a manner that is as efficient as possible compared to what other users might do with a pointing device.

EXAMPLE 1 By browsing a continuously displayed list of currently running windows, the user uses a keyboard to select a window that receives focus.

EXAMPLE 2 By giving a voice command that generates a keyboard-command sequence, the user is able to move focus to any one of several windows.

10.5.4 Enable “always-on-top” windows

Platform software shall enable windows to be set to always remain on top of other windows.

NOTE1 If a function or a window is required continuously for users to perform a task, it is important for the window to be able to be set to always remain visible regardless of its position relative to other windows.

NOTE 2 It is often desirable for a window to remain "always on top" without ever taking focus from other windows, as is discussed in provision 10.5.10 (Enable windows to avoid taking focus).

EXAMPLE 1 The user has a movable on-screen keyboard that is on top of all other windows so that it is visible at all times, but when the user clicks their mouse on the on-screen keyboard another window keeps the input focus and the keyboard input goes to that window.

EXAMPLE 2 A user selects a screen-magnification window that is the top-level window through which all other windows are viewed and which remains always on top.

10.5.5 Provide user control of multiple “always-on-top” windows

Platform software shall provide the user with the option to choose which window is on top or to turn off “always on top”.

NOTE 1 User control is important to prevent a conflict among multiple windows that are specified as “always-on-top.”

NOTE 2 Users may wish to have multiple windows on top of everything else; for example, a calendar and clock. In this case, it may be desirable to provide a facility for users to choose a priority order for multiple “always-on-top” windows.

EXAMPLE Two users each run an on-screen keyboard and a full-screen screen-magnification window. One chooses to run the on-screen keyboard on top of the magnifier and thus at a fixed location on the physical screen, while the other user runs the magnifier on top so that it enlarges the on-screen keyboard.

10.5.6 Enable user choice of effect of input focus on window stacking order

Software should allow users to choose to have the window that receives input focus either automatically placed on top of all other windows (with the exception of an “always-on-top” window, see above) or not have its stacking position changed.

EXAMPLE A user with motor limitations or repetitive-motion injury chooses to move the pointer among windows to automatically bring them to the top rather than to click on them, because it is faster and easier than to click on them.

NOTE Platform software usually handles this functionality as long as applications do not interfere with its normal window handling.

10.5.7 Enable window positioning

Platform software shall give the user the option to adjust the position of all windows, including dialog boxes and software on such systems shall support this option.

NOTE: This helps the users to better use several applications and/or windows at the same time.

EXAMPLE A user with an on-screen keyboard changes the position of a pop-up dialog so that it fits alongside of their keyboard.

10.5.8 Enable window sizing

Platform software should give the user the option to size all windows, including dialog boxes, and software on such systems should support this option.

EXAMPLE User with low vision uses larger font size that causes text to run off the bottom of the default window size. They enlarge the window to see all text.

10.5.9 Support minimise, maximise, restore and close windows

If overlapping windows are supported, software should give the user the option to minimise, maximise, restore and close software windows.

NOTE This helps the users to better use several applications and/or windows at the same time.

EXAMPLE A user who has limited short-term memory clicks on a window’s “Maximum” button in order to see as much of the content as possible.

10.5.10 Enable windows to avoid taking focus

Platform software shall enable windows to avoid taking the focus. The focus should not be assigned to a window that is designated not to accept the focus.

NOTE Any action that would be normally used to reassign the focus to a window that is designated not to accept the focus would not reassign the focus.

EXAMPLE 1 An on-screen keyboard program displays a window containing buttons, and it sets this window to remain "always on top" and to avoid taking the focus. When the user clicks the mouse on a button in this window, the on-screen keyboard sends key events to the application window where the user was working and which still has the focus.

EXAMPLE 2 A user starts a screen-magnification window that is the top-level window through which all other windows are viewed and which remains always on top. When the user clicks anywhere on the screen, the focus remains unchanged and the screen magnifier passes the mouse input to the appropriate underlying window.

10.6 Audio output

10.6.1 Use tone pattern rather than tone value to convey information

When conveying information audibly, software should use temporal or frequency-based tone patterns rather than using a single absolute pitch or volume.

EXAMPLE In a teleconference service, a high to low tone pair (rather than just a low tone) indicates a person signing off.

10.6.2 Enable control of audio volume

Software shall enable users to control the volume of audio output.

10.6.3 Use an appropriate frequency range for non-speech audio

The fundamental frequency of task-relevant non-speech audio used by software should occur in a range between 500 Hz and 3 000 Hz or be easily adjustable by the user into that range [34].

NOTE Sounds in this range are most likely to be detectable by people who are hard of hearing.

10.6.4 Enable adjustment of audio output

Software should enable users to adjust the attributes of task-relevant audio output such as frequency, volume, speed and sound content.

NOTE The range of adjustment will be constrained by the sounds that a system can produce.

EXAMPLE 1 A user can replace the sounds associated with various events and notifications, allowing him or her to choose sounds that he or she is able to distinguish.

EXAMPLE 2 A user may alter the speed of speech from a synthesizer to enhance understanding.

10.6.5 Control of background and other sound tracks

If the background and other sound layers are separate audio tracks/channels,–software should provide a mechanism to enable users to control the volume of and/or turn on/off each audio track.

NOTE Background sounds (e.g., sound effects, music) can mask speech audio or make speech audio more difficult to distinguish by those who are hard of hearing.

EXAMPLE A person who is hard of hearing turns down the background sound so they can understand the dialogue.

10.6.6 Use specified frequency components for audio warnings and alerts

Alerts and other auditory warnings provided by software should include at least two strong mid- to low-frequency components, with recommended ranges of 300 Hz to 750 Hz for one component, and 500 Hz to 3 000 Hz for the other [34].

10.6.7 Allow users to choose visual alternative for audio output

Platform software shall enable users to choose to have task relevant audio output (including alerts) presented in visual form, auditory form or both together ([39], [44] and [46]) and software running on such systems shall support those options.

NOTE This is commonly called ShowSounds (see Annex D).

EXAMPLE 1 By default a beep is provided when an error message has been displayed or a footer message has been updated. For users who have chosen to receive visual feedback, a flashing border on a dialog box is provided in conjunction with a warning tone

EXAMPLE 2 Explanatory text is provided in a dialog box when a distinctive audio (alert or other) is played.

EXAMPLE 3 Software that provides voice output provides closed captions as text that can be displayed on systems providing "closed caption" support or displayed by Braille devices through assistive software.

10.6.8 Synchronise audio equivalents of visual events

Software shall synchronise audible equivalents with the visual events they are associated with.

NOTE 1 This allows a user who cannot see the screen to follow the event sequences.

NOTE 2 Precise synchronisation is not always possible.

10.6.9 Provide speech output services

If the hardware has the ability to support speech synthesis, platform software shall provide programming services for speech output.

NOTE 1 This does not imply that a text to speech engine should always be installed.

NOTE 2 This is relevant for blind users that depend on speech-based assistive technologies.

EXAMPLES SAPI (Speech API) in Microsoft Windows, Java Speech in the Java platform, and Mac OS X TTS on the Apple Macintosh.

10.7 Text equivalents of audio (captions)

10.7.1 Display any captions provided

Software presenting audio information shall provide the facility to display associated captions.

EXAMPLE A media player allows users to display the captions in an “Interactive Tour” which empowers hard of hearing and deaf users to use the tour.

10.7.2 Enable system-wide control of captioning

Platform software should provide a system-wide setting to allow user to indicate that they want available captions to be shown by all software.

NOTE A global setting to enable or disable captions is commonly called ShowSounds and is available on several major platforms (see Annex D).

10.7.3 Support system settings for captioning

Software that presents captions shall use system-wide caption preferences by default. If the system-wide preferences change during playback the new settings shall be used.

EXAMPLE A media player checks the system "ShowSounds" setting when it launches, and displays captions if that value is set to *True*. The media player allows the user to temporarily override this setting but will re-synchronise to the system setting if the system setting changes while it is running.

10.7.4 Position captions to not obscure content

Software that presents captions should position captions to minimize interference with visual content.

EXAMPLE Media player opens up a separate attached window to display captions so they do not cover up the video being played.

10.8 Media and animation

10.8.1 Provide non-animated alternatives to animations

When animation is displayed that will last for more than three seconds, and the user is expected to carry out other activities at the same time, software shall give the user the option to choose an alternative non-animated version of the same content. If the animation is not task relevant then a version where the non-task-relevant item is missing would conform.

EXAMPLE 1 A person with attention deficit disorder chooses a version that does not have animation (or can stop the animation).

EXAMPLE 2 A person with low vision accesses the non-animated version allowing them to see the content without having it move or change before they can view it.

10.8.2 Enable users to stop, start and pause

Software shall enable users to stop, start and pause the presentation.

NOTE “Replay” functions help users to avoid missing some information in the content.

10.8.3 Enable users to replay, rewind, pause and fast or jump forward

Software should enable users to replay, rewind, pause and fast or jump forward the presentation, where appropriate to the task

NOTE “Replay” functions help users to avoid missing some information in the content.

10.8.4 Allow user to control presentation of multiple media streams

Software should enable users to select which media streams are presented, where it is appropriate for the task.

EXAMPLE 1 A user who has both sight and hearing views a captioned video and turns off the audio.

EXAMPLE 2 A user makes a selection to turn off background sound in a video presentation where the voiceover is in a separate media stream from the background sound.

10.8.5 Update equivalent alternatives for media when the media changes

Software should enable equivalent alternatives (e.g., captions, or auditory descriptions of the visual track of a multimedia presentation) to be updated when the content of a media presentation changes (see Checkpoint 6.2 from [7]).

EXAMPLE The audio portion of an “Interactive Tour” video is corrected; the accompanying captions and descriptive audio are corrected at the same time.

10.9 Tactile output

10.9.1 Do not convey information by tactile output alone

Software should not use tactile output alone as the only way to convey information or indicate an action.

NOTE In contrast to visual and acoustic output, only a few sets of symbols are standardized for tactile output (e.g. Braille-code in several versions).

EXAMPLE 1 Bursts of tactile vibrations are verbally described as representing a ringing bell.

EXAMPLE 2 The vibration pattern of a pointing device with tactile feedback is explained independent of the functionality of the pointed object.

EXAMPLE 3 The adjusted maximum level of pressure output of a force feedback system is presented as an alphanumerical value via a visual display.

10.9.2 Use familiar tactile patterns

Software should use well known tactile patterns (familiar in daily life) for presenting tactile messages.

NOTE A person without special knowledge in tactile coding (e.g. like Braille-code, Morse-code etc.) will be mostly well experienced in tactile pattern of daily life

EXAMPLE Bursts of tactile vibrations are designed in analogy to a ringing bell.

10.9.3 Enable tactile output to be adjusted

Software should allow users to adjust tactile output parameters to prevent discomfort, pain or injury.

EXAMPLE A user with reduced haptic perception can individually adjust an upper limit for the tactile output of a force feedback system.

11 Online documentation, Help and support services

11.1 Documentation and Help

11.1.1 Provide understandable documentation and Help

Product documentation and Help for software should be written using a clear and simple language to the extent that this can be done using the vocabulary of the task.

EXAMPLE 1 The documentation of a CAD (Computer Aided Design) system can use terminology from the field of technical drawings.

11.1.2 Provide user documentation in accessible electronic form

All user documentation and help shall be provided in electronic form that meets applicable documentation accessibility standards.

NOTE: The category of “users” includes administrators. For software development software, users would include software developers.

11.1.3 Ensure that on-line documentation and Help is available in accessible forms

Information presented in pictures and graphics by software shall also be provided as descriptive text suitable for screen reading, printing, or Braille conversion so that it can be read by an alternative method ([39], [44] and [46]).

NOTE Using both text and graphics simultaneously (in the default presentation) to communicate information is often helpful to readers who use one to reinforce the other or where people are either visual or verbal in nature.

EXAMPLE A user can print the text portion of the on-line help and read text descriptions of any embedded graphics.

11.1.4 Write instructions and Help without unnecessary device references

Instructions and help for software should be written so that they refer to the users' actions and resulting output without reference to a specific device. References to devices, e.g. the mouse or the keyboard, should only be made when they are integral to and necessary for understanding of the advice being given.

NOTE For contexts where operation of a specific device such as a mouse is required, a generic description may not be possible. However, such specific descriptions need only occur in Help about using that device, not in all contexts.

EXAMPLE 1 The task description in Help does not require a user to recognise the colour of a user interface element to use it, so the text does not state “click on the green icon”. Instead the location as well as the name is reported.

EXAMPLE 2 An application provides a description of how to perform tasks using as many different input/output modalities as are available (e.g. mouse, keyboard, voice, etc.).

11.1.5 Provide documentation and Help on accessibility features

Help or documentation for software shall provide general information on the availability of accessibility features and information about the purpose of and how to use each feature.

NOTE It is important for users to be able to find out about the features that allow the software to be accessible or the features may not be easily discoverable.

EXAMPLE 1 On-line help provides a section describing features of interest for people who have disabilities.

EXAMPLE 2 On-line help explains keyboard-only use of the software.

EXAMPLE 3 On-line help describes how to adjust the font size.

EXAMPLE 4 A product has multiple colour schemes, and documentation and on-line Help describe which colour schemes are appropriate for people with colour vision deficiencies.

11.2 Support services

11.2.1 Provide accessible support services

Technical support and client support services for software shall accommodate the communication needs of end-users with disabilities.

EXAMPLE A company contracts with a relay service to assist the technical support process by providing communication between the company’s support staff and deaf customers who use text telephones.

EXAMPLE 2 Online help or documentation provided on the software company's Web site is designed to comply with published guidelines for making Web content accessible.

11.2.2 Provide accessible training material

If training is provided as part of the product, the training materials should meet applicable accessibility standards.

Annex A
(informative)

Requirements clauses

For the convenience of users of this standard this annex identifies those clauses which contain requirements. The following list identifies the location of the specific guidelines which are required to be addressed when verifying conformance to this part of ISO 9241.

8.2.1 Provide a name for every user interface element

8.2.4 Make names available to assistive technology

8.3.4 Enable individualisation of cursor and pointer

8.3.7 Enable user control of timed responses

8.4.1 Make controls for accessibility features discoverable and operable

8.4.3 Avoid interference with accessibility features

8.4.6 Enable persistent display

8.5.3 Provide accessible system start-up and restart

8.5.4 Enable software-controlled media extraction

8.5.8 Allow warning or error information to persist

8.6.2 Enable communication between software and assistive technology

8.6.3 Use standard accessibility services

8.6.4 Make user interface element information available to assistive technologies

8.6.5 Allow assistive technology to change focus and selection

8.6.6 Provide user interface element descriptions

8.6.7 Make event notification available to assistive technologies

8.6.9 Use system-standard input/output

8.6.10 Enable appropriate presentation of tables

8.6.11 Accept the installation of keyboard and/or pointing device emulators

8.6.12 Allow assistive technology to monitor output operations

8.7.1 Read content on closed systems

8.7.2 Announce changes on closed systems

8.7.3 Operable through tactilely discernable controls

8.7.4 Pass through of system functions

9.1.2 Provide parallel keyboard control of pointer functions

9.2.1 Provide focus cursor

9.2.2 Provide high visibility focus cursor

9.3.2 Enable full use via keyboard

9.3.3 Enable sequential entry of multiple (chorded) keystrokes

9.3.4 Provide adjustment of delay before key acceptance

9.3.5 Provide adjustment of same-key double-strike acceptance

9.3.8 Allow users to turn key repeat off
9.3.12 Reserve accessibility shortcut key assignments
9.3.14 Separate keyboard navigation and activation
9.4.2 Provide direct control of pointer position from external devices
9.4.4 Enable the re-assignment of pointing device button functions
9.4.6 Enable pointing device button-hold functionality
9.4.9 Provide adjustment of multiple-click parameters
9.4.10 Provide adjustment of pointer speed
9.4.11 Provide adjustment of pointer acceleration
9.4.13 Provide a means of finding the pointer
9.4.14 Provide alternatives to simultaneous pointer operations
10.1.1 Avoid seizure-inducing flash rates
10.1.2 Enable user control of time-sensitive presentation of information
10.1.3 Provide accessible alternatives to task relevant audio and video
10.2.4 Provide keyboard access to information displayed outside the physical screen
10.4.1 Do not convey information by colour output alone
10.5.3 Enable non-pointer navigation directly to windows
10.5.4 Enable “always-on-top” windows
10.5.5 Provide user control of multiple “always-on-top” windows
10.5.7 Enable window positioning
10.5.10 Enable windows from always taking focus
10.6.2 Enable control of audio volume
10.6.7 Allow users to choose visual alternative for audio output
10.6.8 Synchronise audio equivalents of visual events
10.6.9 Provide speech output services
10.7.1 Display any captions provided
10.7.3 Support system settings for captioning
10.8.1 Provide non-animated alternatives to animations
10.8.2 Enable users to stop, start and pause
10.8.5 Update equivalent alternatives for media when the media changes
11.1.2 Provide user documentation in electronic form
11.1.3 Ensure that on-line documentation and Help is available in accessible forms
11.1.5 Provide documentation and Help on accessibility features
11.2.1 Provide accessible support services

Annex B
(informative)

Sample procedure for assessing applicability and conformance

B.1 General

This Annex provides an example of a checklist that can be used to determine whether the applicable requirements and recommendations in this part of ISO 9241 have been met.

The checklist can be used either during product development or for evaluation of a completed product.

The checklist contains all requirements and recommendations from the standard, presented in sequence.

It should be noted that the procedure described is provided as guidance and is not an exhaustive process to be used as a substitute for the standard itself. The use of the checklist provides a basis for:

— determining which of the requirements and recommendations are applicable,

— determining whether applicable requirements and recommendations have been adhered to,

— providing a list in support of a claim of conformance that shows that all applicable requirements have been adhered to, and a systematic listing of all the applicable recommendations that have been adhered to.

The majority of the requirements are applicable to all software if it is to enable use by people with the widest possible range of capabilities. However, in some circumstances what is needed to make the software accessible depends upon the context of use (the users, the tasks, the environment, and the technology). Where a conditional 'if' appears in either a requirement or a recommendation it is necessary to determine whether or not the context of use in which the software is, or is intended to be, used, is included within the conditions covered by the 'if' statement. For each context-dependent requirement or recommendation, information on applicable circumstances is given in the clause. If the conditional statement does not apply and thus the requirement or recommendation is not applicable, this should be noted in the relevant column in the applicability section of the table, and a brief explanation should be provided in the column 'Reason not applicable'.

The next step involves determining whether the software being evaluated conforms to each requirement or recommendation (where applicable). The exact method for making this decision could vary from an inspection-based judgment of whether a feature is or is not present to testing the software with users. Whatever the method of evaluation considered most appropriate, the checklist provides space to give an indication of the level of conformity, and comments on the method used or the judgment made under the 'Comments' column.

The completed checklist can be used in support of statements relating to conformance of software with the standard, providing a list of all applicable requirements being conformed to, and a list of which of the applicable recommendations are being conformed to.

B.2 How to use the Table

Clause numbers and titles are presented in the first two columns of the table.

The third column is used to indicate whether the requirement or recommendation in each clause is applicable or not applicable. All those requirements that have no conditions attached to them have already have Y

inserted in the third column to show that they are applicable. C indicates that they are applicable unless the specified conditions apply.

All the other clauses need to be checked in relation to the design context of the specific software system being developed or assessed. It should be noted that some requirements for which there is a conditional statement will need to have column three completed.

In addition the applicability of all the recommendations should be checked and Y or N entered in column three, as appropriate.

Where a requirement or recommendation is not applicable a brief note giving the reasons should be inserted in column four.

When checking whether a requirement or recommendation has been satisfied it will be necessary to review all those items which are shown to be applicable in column 3.

There should be an entry in the relevant place in column five, six or seven, showing whether each applicable requirement or recommendation has been satisfied (Yes), partially satisfied (Partial) or not satisfied (No). Any clause which is either judged to be partially satisfied, or not satisfied should be accompanied by a brief note explaining the reasons why this is the case.

Table B.1

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
8	**General Guidelines**						
8.1	**Input/output alternatives**						
8.1.1	Enable user input/output choice						
8.1.2	Enable switching of input/output alternatives						
8.2	**Names for user interface elements**						
8.2.1	Provide a name for every user interface element	**Y**					
8.2.2	Provide meaningful names						
8.2.3	Names provided by the system should be unique						
8.2.4	Make names available to assistive technology	**Y**					
8.2.5	Display names						
8.2.6	Provide short names and labels						
8.2.7	Provide text label display option for icons						
8.2.8	Properly position the labels of user interface elements on the screen						

Table B.1 *(continued)*

Clause no.	Guideline	Applicability					
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
8.3	User preference settings						
8.3.1	Enable easy individualisation of user preference settings						
8.3.2	Enable adjustment of common user interface elements						
8.3.4	Enable individualisation of cursor and pointer	**C**					
8.3.5	Provide user-preference profiles						
8.3.6	Provide capability to use preference settings across locations						
8.3.7	Enable user control of timed responses	**C**					
8.4	**Special considerations for accessibility adjustments**						
8.4.1	Make controls for accessibility features discernable and operable	**Y**					
8.4.2	Safeguard against inadvertent activation or deactivation of accessibility features						
8.4.3	Avoid interference with accessibility features	**Y**					
8.4.4	Inform user of accessibility feature On/Off status						
8.4.5	Inform user of accessibility feature activation						
8.4.6	Enable persistent display	**C**					

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
8.5	**General control and operation guidelines**						
8.5.1	Optimise the number of steps required for any task						
8.5.3	Provide accessible system start-up and restart	**C**					
8.5.4	Enable software-controlled media extraction	**C**					
8.5.5	Support “Copy” and “Paste” operations						
8.5.6	Support “Copy” in non-editable text						
8.5.7	Enable selection of elements as an alternative to typing						
8.5.8	Allow warning or error information to persist	**Y**					
8.5.9	Present user notification using consistent presentation techniques						
8.5.10	Provide understandable user notifications						
8.6	**Compatibility with assistive technology**						
8.6.1	General						
8.6.2	Enable communication between software and assistive technology	**Y**					
8.6.3	Use standard accessibility services	**Y**					

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
8.6.4	Make user interface element information available to assistive technologies	**Y**					
8.6.5	Allow assistive technology to change focus and selection attributes	**Y**					
8.6.6	Provide user interface element descriptions	**C**					
8.6.7	Make event notification available to assistive technologies	**Y**					
8.6.8	Allow assistive technology to access resources						
8.6.9	Use system-standard input/output	**C**					
8.6.10	Enable appropriate presentation of tables	**Y**					
8.6.11	Accept the installation of keyboard and/or pointing device emulators	**Y**					
8.6.12	Allow assistive technologies to monitor output operations	**Y**					
8.7	**Closed systems**						
8.7.1	Read content on closed systems	**Y**					
8.7.2	Announce changes on closed systems	**Y**					
8.7.3	Operable through tactilely discernable controls	**Y**					
8.7.4	Pass through of system functions	**Y**					

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
9	**Inputs**						
9.1	**Alternative input options**						
9.1.1	Provide keyboard input from all standard input mechanisms						
9.1.2	Provide parallel keyboard control of pointer functions	**Y**					
9.1.3	Provide pointer control of keyboard functions						
9.1.4	Provide speech recognition services						
9.2	**Keyboard focus**						
9.2.1	Provide focus cursor	**Y**					
9.2.2	Provide high visibility focus cursor	**Y**					
9.2.3	Restore state when regaining focus	**Y**					
9.3	**Keyboard input**						
9.3.1	General						
9.3.2	Enable full use via keyboard	**C**					
9.3.3	Enable sequential entry of multiple (chorded) keystrokes	**Y**					
9.3.4	Provide adjustment of delay before key acceptance	**Y**					
9.1.5	Provide adjustment of same-key double-strike acceptance	**Y**					
9.3.6	Provide adjustment of key repeat rate						

Table B.1 *(continued)*

Clause no.	Cuideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
9.3.7	Provide adjustment of key repeat onset						
9.3.8	Allow users to turn key repeat off	**Y**					
9.3.9	Provide notification about toggle-key status						
9.3.10	Provide accelerator keys						
9.3.11	Provide implicit designators						
9.3.12	Reserve accessibility short cut key assignments	**Y**					
9.3.13	Enable remapping of keyboard functions						
9.3.14	Separate keyboard navigation and activation	**Y**					
9.3.15	Follow operating system keyboard conventions						
9.3.16	Facilitate long list and menu navigation						
9.3.17	Arrange input/output controls in task-appropriate groups						
9.4	**Pointing devices**						
9.4.1	General						
9.4.2	Provide direct control of pointer position from external devices	**Y**					
9.4.3	Provide easily-selectable pointing device targets						

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
9.4.4	Enable the re-assignment of pointing device button functions	**Y**					
9.4.5	Provide alternative input methods for complex pointing device operations						
9.4.6	Enable pointing device button-hold functionality	**Y**					
9.47	Provide adjustment of delay of pointer-button-press acceptance						
9.4.8	Provide adjustment of minimum drag distance						
9.4.9	Provide adjustment of multiple-click parameters	**Y**					
9.4.10	Provide adjustment of pointer speed	**Y**					
9.4.11	Provide adjustment of pointer movement acceleration	**C**					
9.4.12	Provide adjustment of pointer movement direction						
9.4.13	Provide a means of finding the pointer	**C**					
9.4.14	Provide alternatives to simultaneous pointer operations	**Y**					
10	**Outputs**						
10.1	**General output guidelines**						
10.1.1	Avoid seizure-inducing flash rates	**Y**					
10.1.2	Enable user control of time-sensitive presentation of information	**Y**					

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
10.1.3	Provide accessible alternatives to task relevant audio and video	**C**					
10.2	**Visual output (displays)**						
10.2.1	Enable users to adjust graphic attributes						
10.2.2	Provide a visual information mode usable by users with low visual acuity						
10.2.3	Use text characters as text, not as drawing elements						
10.2.4	A Provide keyboard access to information displayed outside the physical screen	**C**					
10.3	**Text/Fonts**						
10.3.1	Enable users to set minimum font size						
10.3.2	Adjust the scale and layout of user interface elements as font-size changes						
10.4	**Colour**						
10.4.1	Do not convey information by colour output alone	**Y**					
10.4.2	Provide colour schemes for people who have visual impairments						
10.4.3	Provide individualisation of colour schemes						
10.4.4	Allow users to individualise colour coding						

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
10.4.5	Provide contrast between foreground and background						
10.5	**Window appearance and behaviour**						
10.5.1	Provide meaningful window titles						
10.5.2	Provide unique window titles						
10.5.3	Enable non-pointer navigation directly to windows	**Y**					
10.5.4	Enable “always-on-top” windows	**Y**					
10.5.5	Provide user control of multiple “always-on-top” windows	**Y**					
10.5.6	Enable user choice of effect of input focus on window stacking order						
10.5.7	Enable window positioning	**Y**					
10.5.8	Enable window sizing						
10.5.9	Support minimise, maximise, restore and close windows						
10.5.10	Enable windows to avoid taking focus	**Y**					
10.6	**Audio output**						
10.6.1	Use tone pattern rather than tone value to convey information						
10.6.2	Enable control of audio volume	**Y**					
10.6.3	Use an appropriate frequency range for non-speech audio						
10.6.4	Enable adjustment of audio output						

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
10.6.5	Control of background and other sound tracks						
10.6.6	Use specified frequency components for audio warnings and alerts						
10.6.7	Allow users to choose visual alternative for audio output	**Y**					
10.6.8	Synchronise audio equivalents of visual events	**Y**					
10.6.9	Provide speech output services	**Y**					
10.7	**Text equivalents of audio (captions)**						
10.7.1	Display any captions provided	**Y**					
10.7.2	Enable system-wide control of captioning						
10.7.3	Support system settings for captioning	**Y**					
10.7.4	Position captions to not obscure content						
10.8	**Media and animation**						
10.8.1	Provide non-animated alternatives to animations	**Y**					
10.8.2	Enable users to stop, start and pause	**Y**					
10.8.3	Enable users to replay, rewind, pause and fast or jump forward						
10.8.4	Allow user to control presentation of multiple media streams						

Table B.1 *(continued)*

Clause no.	Guideline	Applicability		Conformance			
		Yes or No	Reason not applicable	Yes	Par-tial	No	Comments
10.8.5	Update equivalent alternatives for media when the media changes	**C**					
10.9	**Tactile output**						
10.9.1	Do not convey information by tactile output alone						
10.9.2	Use familiar tactile patterns						
10.9.3	Enable tactile output to be adjusted						
11	**Online documentation, Help and support services**						
11.1	**Documentation and Help**						
11.1.1	Provide understandable documentation and Help						
11.1.2	Provide documentation in accessible electronic form	**Y**					
11.1.3	Ensure that on-line documentation and Help is available in accessible forms	**Y**					
11.1.4	Write instructions and help without unnecessary device references						
11.1.5	Provide documentation and help on accessibility features	**Y**					
11.2	**Support services**						
11.2.1	Provide accessible support services	**Y**					
11.2.2	Provide accessible training material						

Annex C
(informative)

Issues regarding activity limitations

C.1 General

This Annex provides some additional information about sources of limitation on typical activities involving the use of software systems, and their implications for designing for accessibility.

While these sources of limitation are frequently described in relation to underlying body functions as used in the World Health Organisation International Classification of Functioning, Disability and Health (ICF), (WHO 2002), the same limitations may arise from other sources, such as the particular context in which individuals find themselves at any time.

For the purposes of this Annex, three main area of the ICF classification of Body Functions are considered most relevant to interaction with software systems; sensory functions, including seeing and hearing; neuromusculoskeletal and movement related functions; and mental functions, including attention, memory and language. In addition reference is made to the implications for accessibility of combined sources of limitation due to body function.

C.2 Sensory Functions

C.2.1 Vision

For many interactive software systems a major part of the interaction between the user and the technology relies on the use of visually presented material.

C.2.1.1 Individuals who are unable to see

For any user who is unable to see this means that other senses will have to be used and appropriate provision made to enable access to equivalent content and resources via those senses. In addition, individuals may have normal vision but are unable to view a screen due to context or task-related issues. For example: whilst driving a car a motorist is unable to view the screen of their GPS system.

Typical non-visual forms of interface used in interactive software are auditory or tactile. Whether used as a substitute for visual interfaces or in their own right, the primary issues are how to:

— obtain information provided by sound or tactile display whether or not connected with a visual presentation,

— navigate in an auditory or tactile environment, and/or achieve equivalent navigation to that among elements presented visually,

— identify user interface elements, and

— control focus, navigation, and other functions via the keyboard, joy stick, voice or other control actuator.

Some individuals who cannot see will use specialised assistive technologies. For example, individuals who have learned Braille can take advantage of software and hardware that will provide ‘screen readers’ which will produce Braille output. Those who become blind later in life are less likely to learn such specialised skills

although they may learn some new auditory skills and thus rely on additional auditory methods to obtain information

"Screen readers", i.e. assistive software that can provide spoken information for windows, controls, menus, images, text and other information typically displayed visually on a screen, help people who have to rely main on speech to convey information. Others use tactile displays such as dynamic Braille or Moon displays

Interactions based on spatial relationships and the use of visual metaphors present users who cannot see with the greatest difficulties in terms of the provision of equivalent information in another modality. It is important therefore that all information (not just text) that is provided visually be available to assistive technologies for alternate display.

In addition, the use of speech output to substitute for visually presented material may cause problems due to the potential difficulty of attending to other auditory outputs that occur while they are listening. Braille and other tactile displays can assist here.

C.2.1.2 Individuals with low vision

Persons with low vision often use technologies more commonly associated with those who are unable to see (e.g., screen readers). However, for individuals with low vision it is important to find ways to facilitate their use of their remaining vision whenever possible. Sight, even limited sight is a very powerful capability and they should not have to fall back and access things as if they were blind. Combinations of visual and auditory techniques are often most effective. (Tactile can sometimes be used but is less common except for individuals with very low vision.

It is a universal experience that vision changes throughout life and once adulthood is reached vision tends to become less effective over time. In addition a variety of factors such as low acuity, colour-perception deficits, impaired contrast sensitivity, loss of depth perception, and restricted field of view may affect the ability of individuals to see and discriminate visually presented material. Environmental factors such as glare from sunlight, or light sources with poor colour rendering, may have similar consequences. An individual who has reduced acuity may find that ordinary text is often difficult to read, even with the best possible correction.

The main approach in terms of increasing accessibility (other than by removing externally generated sources of interference with vision) is to provide means by which the visually presented material can be changed to increase its visibility and discriminability. Individuals interacting with systems in low vision conditions may experience particular difficulties in detecting size coding. They may experience difficulties with font discrimination and with locating or tracking user interface elements such as pointers, cursors, drop targets, hot spots and direct manipulation handles.

Support consists of the provision of means for increasing the size, contrast and overall visibility of visually displayed material, as well as allowing choice of colours and colour combinations. What is required in any case depends upon an individual's specific visual needs and thus depends upon the capacity for individualisation. Common assistive technologies include the use of oversized monitors, large fonts, high contrast, and hardware or software magnification to enlarge portions of the display.

Additionally, non-visual or low visual conditions may cause difficulties when very small displays, such as those on printers, copiers and ticket machines are required to be read.

C.2.2 Hearing

C.2.2.1 Individuals who are unable to hear

Individuals may be unable to hear sound and thus be unable to detect or discriminate information presented in auditory form. The inability to hear sound below 90 dB is generally taken as the criterion for an individual being unable to hear. Disabling environments may occur when individuals cannot hear signals generated by the system, for example if there is a very high ambient noise level or the use of hearing protection.. These situations must be regarded as creating limitations on the ability of individuals to use the system. In these circumstances the preferred solution is to eliminate the source of the problem. However, where this may be

impractical, the approach will be to implement the same software based solutions that are appropriate for individuals who cannot hear in standard environments.

When interacting with software systems, users who cannot hear will encounter problems if important information is presented only in audio form. It is therefore important to enable the presentation of auditory information in other modalities. For example, verbal information can be provided by common symbols, text format or the “ShowSounds” feature, that notifies software to present audio information in visual form. These techniques will also be of benefit to individuals in contexts where sound is masked by background noise (e.g. a machine shop floor) or where sound is turned off or cannot be used (e.g. a library).

Some individuals with a general inability to detect auditory information may also experience limitations on voice and speech functions. This may have implications for their ability to produce speech recognisable by voice-input systems and should be considered when such technology is being implemented. In addition, if their experience of a national language is as a second language (sign language often being the primary language for people who become deaf at an early age or who are born with deafness) this will have implications for the form of language used in the presentation of visual alternatives as a consequence of the learning aspects of mental function.

Some individuals who are deaf interact with software, such as interactive voice response systems, via a telecommunication device for the deaf (TDD), a text telephone (TTY), or a relay service in which a human operator types spoken text (e.g., IVR prompts) and relays it to the user. It is important that designers ensure that applications like this are accessible to these users and do not impose unnecessary response time requirements that render transactions impossible.

C.2.2.2 Individuals with a reduced ability to hear

Individuals may experience difficulties in hearing and discriminating auditory information both as a result of individual capabilities and as a result of external sources of interference. Issues that may arise include:

— the inability to detect sound

— the inability to discriminate frequency changes, differential decreases in sensitivity across the frequency range, and select frequency ranges where they have low sensitivity;

— difficulty in localising sounds;

— difficulty in discerning sounds against background noise

— inappropriate response due to mishearing or not hearing auditory information

As with individuals who are unable to hear, the main implications for accessibility involve the provision of equivalent versions of auditory material via another modality, for example the use of the “ShowSounds” feature that notifies software to present audio-information in visual form. In addition, individuals with a reduced ability to hear may adjust auditory material through the ability to individualise the characteristics of auditory presentations (e.g increasing volume or selectively changing the frequency spectrum used).

Individuals with a reduced ability to hear may or may not use hearing aids, but to the extent that this form of assistive technology can take advantage of selective auditory inputs from the software system, the availability of such input will increase accessibility. It is very common for individuals with a reduced ability to hear to use whatever hearing they have and thus combine modalities (e.g., use captions as well as audio).

Finally, as with individuals who are unable to hear, some users with a reduced ability to hear may experience limitations on voice and speech functions. This may have implications for their ability to produce speech recognisable by voice-input systems and should be considered when such technology is being implemented.

C.2.3 Tactile

Some individuals, due to disease, accident or aging, have reduced tactile sensations. This can interfere with any tactile output modes. The ability to have information available in a variety of forms is important to address this. It is also important to not make assumptions about alternate modalities that may be available since they may not work for some users. For example, diabetes can cause loss of visions and loss of sensation in the fingers.

C.3 Neuromusculoskeletal and movement related functions

C.3.1 General

Interaction with software systems is highly dependent on the means of input/output used by individuals. While general mobility of the whole body may not be a critical factor, motor activity of the limbs and hands whether affected by the mobility and stability of joint and bone functions, the power, tone and endurance functions of the muscles, or the voluntary or involuntary control of movement, is critical to successful interaction. The design of software must take account of the range and variety of characteristics that may be present within the user population.

C.3.2 Individuals with limitations in motor activity

There are many factors that may influence motor activity. The causes of limitations on activity may be impairments that are long term and/or progressive, or temporary, as well as contextually determined, for example the need to carry out another task while interacting with the software. Particular issues are that individuals may have poor co-ordination abilities, weakness, difficulty reaching, involuntary movement (e.g., tremor, athetosis), and problems in moving a body part. Pain and soft tissue injuries can also cause limitations in a person's physical abilities causing them to have to use alternate means for input. Individuals, as a simple consequence of the aging process, experience a slowing of reaction time and speed of motor actions. Designers need to ensure that applications intended for older adults take this into account and allow sufficient time for user actions in software that requires timed responses.

Individuals with limitations on motor activity may or may not use assistive technologies. There is a wide variety of hardware and software that may be employed by those who do and therefore it is not possible to describe the full range in detail here. Examples include eye-tracking devices, on-screen keyboards, speech recognition, and alternative pointing devices.

The extreme variation in the needs and capabilities of individuals who have motor activity limitations makes the provision of the capacity for individualisation of input parameters critical to achieving accessibility. In particular it is necessary to be able to customise input parameters in terms of spatial allocations of functionality and the timing that underpins interaction.

C.3.3 Physical Stature

Some individuals are of small stature. This may be because they are children or because it is their adult size. This can cause problems with reach, hand size, normal seated position etc. In either case most access issues are hardware in nature. However, the ability to use alternate input devices may also be very important to this group.

C.3.4 Speech Disabilities

Speech is also a form of motor activity, and some individuals have impairments that make their speech difficult to understand, or they may not be able to speak at all. Individuals may experience a speech disability for any one of a number of reasons. They may have an impairment or injury to the physical mechanisms of speech (respiration, vocal tract, physical manipulators). They may have problems in the neuromuscular systems that control speech. They may have cognitive impairments of the speech or language centres. User interfaces that include speech input need to provide other input options that can substitute for speech, or be able to utilize the

speech of these individuals, some of whom may be using augmentative communication devices to produce speech.

C.4 Mental functions

C.4.1 General

Variation in psychological functioning probably represents greater diversity than in any other function in human beings. Of particular concern to software accessibility is the area of cognitive functioning that relates to the handling of information. Receiving information, processing it and making appropriate responses forms the major element in the use of interactive software systems. These human cognitive capabilities are very diverse, highly adaptable and subject to change throughout life.

The issues commonly encountered by users who have cognitive disabilities involve difficulties receiving information, processing it and communicating what they know. Users with these impairments may have trouble learning, making generalisations and associations and expressing themselves through spoken or written language. Attention-deficit hyperactivity disorders make it difficult for a person to sit calmly and give full attention to a task.

The issues commonly faced by users who have dyslexia are that they have difficulty reading text that is presented in written form and producing written text.

While there are issues that are well understood, and for which it is possible to provide specific guidance, it has to be recognised that understanding of cognitive function is incomplete and that individuals vary to such an extent that no general guidance is appropriate in many situations. Much of the guidance in ergonomics standards relating to the design of dialogue interfaces, in particular ISO 9241-10 and -12 to -17, is based on the currently established understanding of human cognitive function.

C.4.2 Limitations on attention

For many information handling tasks it may be necessary to focus attention, or sustain attention over a period of time. Strategies for helping to identify the required focus of attention involving formatting and presentation of information will be beneficial. For people with limits on their capacity to sustain attention it is important to provide the capacity to adjust the characteristics of non task relevant displayed information, to avoid potential distraction**.**

C.4.3 Limitations on memory

Information handling tasks are very sensitive to limitations on both short and long term memory. In particular people experience problems recalling information from long term memory and will have difficulties holding newly presented information in short term memory if there is too much of it or if it has to be retained for too long. Thus the design of the interactive software should enable recognition rather than demanding recall, wherever possible, and use information consistently and in a way that is consistent with user expectations. Demands on short term memory should be minimised.

C.4.4 Limitations on the mental functions of language

Limitations in the ability to understand and produce written language have implications for the ways language is used in software systems. These limitations may result from a wide variety of causes including conditions at birth, illness, accident, medical procedures to address other conditions and aging. They may be physical, cognitive or psychological in nature. Individuals with reduced auditory perception or deafness, and who rely on visual communication may sometimes also have reduced language skills in a printed language (that is different than their primary visual language). Regardless of cause, they result in reduced ability to handle written and/or spoken language. Standards providing guidance on clear expression and presentation of language will be important to achieve ease of reading for the largest possible number of readers. In addition specific options to provide additional support, such as providing the option to have an auditory version of the

text available in parallel, will be beneficial. For individuals with limitations on writing ability, alternate inputs as well as use of symbols and speech output can be of help. Support within software for alternate input and output compatibility is therefore important.

C.5 Individuals with other disabilities.

C.5.1 Balance

Some individuals may have disabilities that affect their sense of touch, their haptic sense, the functioning of their vestibular system (sense of balance). and sense of smell or taste. Although these abilities are infrequently required for the use of software user interfaces, systems designed to utilize these senses need to take into account the implications of disabilities related to these senses and provide alternative input and output channels to enable users affected by these disabilities to user the software.*)*

C.5.2 Allergy

Some individuals experience allergic reactions to chemicals or other substances that are so severe that it prevents them from breathing or being in contact with them for any extended period of time. This can cause severe limitations in the environments that they can live or work in or in the materials that devices they use can be made of. This is largely a hardware problem however that doesn't affect the design of software. However that ability for software to accept alternate inputs and to allow variations in display can help for some.

C.6 Multiple body function effects

Individuals may experience limitations on function in more than one of the areas of body function at the same time and this creates greater complexity in terms of achieving accessibility in interactive software systems. This may particularly occur with increasing age when changes in sensory, motor and mental function may occur in parallel. Experience of the effects of age on body function is universal and as such is a matter of concern for older users. For this reason the greater the possibility of integrating design solutions that address the full range of user capabilities within the software system, rather than requiring add-on assistive technologies, the greater the positive outcome in terms of achieving accessibility and removal of potential sources of stigma.

As noted with respect to motor functions, slowing also occurs with respect to cognitive functions as people age. When the user population contains people who experience cognitive slowing, whether age-related or due to some other condition, developers of software need to ensure that users have sufficient time to complete activities that may have time constraints imposed on their execution.

Combinations of such limitations may mean that some of the guidance offered may not address the needs of individuals who are exposed to such combined effects. For example auditory output of visually presented text may not be of any help to somebody who is not only experiencing low vision but also has loss of hearing. The complex interactions that arise from these different sources increase the demand to ensure that solutions can be individualised to meet specific needs.

Annex D
(informative)

StickyKeys, SlowKeys, BounceKeys, FilterKeys, MouseKeys, RepeatKeys, ToggleKeys, SoundSentry, ShowSounds, Time Out and SerialKeys

(Computer Interface Access Enhancements developed by the Trace R&D Center, University of Wisconsin-Madison, USA)

This information is given for the convenience of users of this International Standard and does not constitute an endorsement by ISO.

D.1 Introduction

This set of access features was developed by the Trace R&D Center at the University of Wisconsin-Madison to make computer systems usable by people with a wide range of physical and sensory impairments. Implementations of these access features are available for at least eight operating systems and environments including: Apple IIe, IIgs, and MacOS (by Trace R&D Center and Apple); IBM PC-DOS and Microsoft DOS (AccessDOS by Trace Center for IBM); X Windows (Access X by the X-Consortium); Microsoft Windows 2.0 and 3.1 (Access Pack by Trace R&D Center); Windows 95, 98, NT, ME, XP and VISTA (by Microsoft); and Linux (currently under implementation).

Source code and prototype implementations for these access features have been recently released under an open source license that allows incorporation into commercial products.

D.2 Permission to Use Terms

The terms StickyKeys™, SlowKeys™, BounceKeys™, FilterKeys™, MouseKeys™, RepeatKeys™, ToggleKeys™, SoundSentry™ ShowSounds™ and SerialKeys™ are all trademarks of the University of Wisconsin. However, use of the terms is permitted freely without royalty or license to describe user interface features that have the functionality and behaviour described below.

D.3 Description of Access Features

NOTE There are two methods for turning many (but not all) of these features on and off. One method is through the control panel. A second method is to turn them on and off directly using keyboard shortcuts. The keyboard shortcuts are provided for two reasons. One – these features may be needed on kiosks and other closed systems where the control panels are not available. Two – for some users and operating systems, it may be difficult to open the control panels unless the feature is turned on. Where both methods are defined they are described below.

When turning the features on from the keyboard a low-high slide sound should be emitted. A high-low slide sound should be used for Off).

D.3.1 StickyKeys

StickyKeys is designed for people who cannot use both hands, or who use a dowel or stick to type. StickyKeys allows individuals to press key combinations (e.g. CTRL-ALT-DEL) sequentially rather than having to hold them all down together. StickyKeys works with those keys defined as "modifier" keys, such as the Shift, Alt and Control keys. Usually the StickyKeys status is shown on-screen at the user’s option.

D.3.1.1 Turn feature On:

Two methods, both essential:

— from the control panel and

— from the keyboard by pressing the Shift key 5 times with no intervening key presses or mouse clicks . A confirmation dialog is recommended for keyboard activation of this feature, though the user should be able to turn this dialog off. Audible and visual indicators are recommended when the feature is turned on or off.

NOTE The keyboard activation feature may be enabled/disabled in the control panel.

D.3.1.2 Turn feature Off:

Using the same two methods as Turn On, plus (at user's option) turn off anytime two keys are pressed simultaneously. Audible and visual indicators are recommended when the feature is turned On or Off.

D.3.1.3 Operation:

— Pressing and releasing any 'modifier' key once causes a low-high tone and causes that modifier key to "latch" so that the next (single) non-modifier key pressed (or the next pointing device button action) is modified by the latched 'modifier' key(s).

— Pressing any modifier key twice sequentially causes a high tone and 'locks' that modifier key down. All subsequent non-modifier keys pressed, pointing device actions, and any software actions that are altered by modifier key state are modified by the locked modifier key(s).

— Pressing a 'locked' modifier key once unlocks and releases it and causes a low tone.

NOTE Multiple modifier keys can be latched and/or locked independently.

D.3.1.4 Adjustments:

— On/Off - StickyKeys (default is Off),

— On/Off - Keyboard Shortcut - (5 shift key activations) (default is On),

— On/Off - Keyboard Activation Confirmation Dialog (default is On),

— On/Off - Auto-turnoff if two keys held down (default is On),

— On/Off - On-screen StickyKeys status indicators (default is On),

— On/Off - Audible indication of StickyKeys activation and use (optional) (default is On).

NOTE The system should provide the StickyKeys feature for all standard modifier keys. Some modifier keys, such as the Fn key found on many notebook computers, may need to have StickyKeys implemented in the keyboard firmware.

D.3.2 SlowKeys

SlowKeys is designed for users who have extra, uncontrolled movements that cause them to strike surrounding keys unintentionally when typing. SlowKeys causes the keyboard to ignore all keys that are bumped or pressed briefly. Keystrokes are accepted only if keys are held down for a user specifiable period of time.

D.3.2.1 Turn feature On:

Two methods, both essential:

— from the control panel and

— from the keyboard by holding the right shift key down for 8 s (if keyboard activation of this feature has been enabled in the control panel). A confirmation dialog is recommended for keyboard activation of this feature, though the user should be able to turn this dialog off. Audible and visual indicators are recommended when the feature is turned on or off.

D.3.2.2 Turn feature Off:

Using the same two methods as Turn On, plus reboot. (Feature is always off at boot time because it makes the keyboard look like it is broken. Rebooting must turn feature off.)

D.3.2.3 Operation:

— Pressing the right shift for 8 s causes the feature to Turn On. A double beep is emitted at 5 s to cause any inadvertent holding of the shift key to be stopped. This “right-shift-key” start must be enabled from the control panel.

— Once SlowKeys is turned On, the keyboard does not accept any keystrokes unless keys are held down for the SlowKeys acceptance time. When a key is first pressed a high tone is emitted. After the preset “acceptance” time has elapsed a second low tone is emitted and the key stroke is accepted (key down is sent).

D.3.2.4 Adjustments:

— On/Off - SlowKeys (default is Off),

— On/Off - Keyboard activation & deactivation of SlowKeys (default is On),

— On/Off - Keyboard activation confirmation dialog (default is On),

— On/Off - Audible indication of SlowKeys activation and use (optional) (default is On).

— Delay-before-acceptance setting for SlowKeys (a minimum range of 0,5 s to 2 s, default is 0,75 s).

NOTE It is acceptable to make SlowKeys and BounceKeys be mutually exclusive, however both of these features can be active at the same time.

D.3.3 BounceKeys

BounceKeys is designed for users with tremor that causes them to inadvertently strike a key extra times when pressing or releasing the key. BounceKeys only accepts a single keystroke at a time from a key. Once a key is released it will not accept another stroke of the same key until a (user settable) period of time has passed. BounceKeys has no effect on how quickly a person can type a different key.

D.3.3.1 Turn feature On:

Two methods, both essential:

— from the control panel and

— from the keyboard by holding the right shift key down for 8 seconds (if keyboard activation of this feature has been enabled in the control panel). A confirmation dialog is recommended for keyboard activation of

this feature, though the user should be able to turn this dialog off. Audible and visual indicators are recommended when the feature is turned on or off.

D.3.3.2 Turn feature Off:

Using the same two methods as Turn On, plus reboot (feature is not On at boot time or is not on at boot time if delay is longer than 0,35 s.)

D.3.3.3 Operation:

Once turned On the user types as usual at full speed. Any rattling of keys will be ignored. To type two of the same letter in a row, the user simply waits briefly between key-presses (usually half a second or so).

D.3.3.4 Adjustments:

— On/Off - BounceKeys (default is Off),

— On/Off - keyboard activation & deactivation of BounceKeys (default is Off),

— On/Off - keyboard activation confirmation dialog (default is On),

— On/Off - audible indication of BounceKeys activation and use (optional) (default is On).

— BounceKeys debounce delay before accepting the same key again (a minimum range of 0,2 s to 1 s) (default is 0,5 s).

NOTE It is acceptable to make SlowKeys and BounceKeys be mutually exclusive, however both of these features can be active at the same time.

D.3.4 FilterKeys

A name sometime used for the BounceKeys and SlowKeys features packaged together. It is acceptable to make these two features be mutually exclusive, however they can both be active at the same time.

D.3.5 MouseKeys

MouseKeys is designed for users who cannot use a mouse accurately (or at all) because of physical capability. MouseKeys allows the individual to use the keys on the numeric keypad to control the mouse cursor on screen and to operate the mouse buttons.

D.3.5.1 Turn feature On:

Two methods, both essential:

— from the control panel and

— from the keyboard by pressing the key combination specified by the OS (if keyboard activation of this feature has been enabled in the control panel).

NOTE New OS implementations should consider using LeftShift-LeftAlt-NumLock key combination.

D.3.5.2 Turn feature Off:

— Using the same two methods as Turn On.

D.3.5.3 Operation:

Once MouseKeys is turned On the NumLock key can be used to switch the keypad back and forth between MouseKeys operation and one of the other two standard modes of keypad operation (number pad or key navigation). It is recommended that there be an option for showing MouseKeys status on screen.

When in MouseKeys mode the keypad keys operate in the following fashion:

Controlling pointer movement (For computers with a number pad):

1 – Move down and to the left
2 – Move down
3 – Move down and to the right
4 – Move to the left
6 – Move to the right
7 – Move up and to the left
8 – Move up
9 – Move up and to the right

NOTE 1 With all of these movement keys a press and release moves the pointer by one pixel.

NOTE 2 Pressing and holding a key down causes the pointer to move by one pixel and then after a pause of 0,5 s it begins to move and accelerate.

NOTE 3 If the Control key is held down, a press and release causes the pointer to jump by multiple pixels (e.g. 20 pixels). (Optional)

NOTE 4 If the Shift key is held down, the mouse moves by one pixel only no matter how long the movement key is held down. (Optional).

Clicking and dragging (Recommended) :

5 – Click the selected mouse button
+ – Double-click the selected mouse button
. – Lock down the selected mouse button
0 – Release all locked mouse buttons

Selecting mouse buttons (Recommended) :

/ – Select the left mouse button to be controlled with MouseKeys
* – Select the center mouse button to be controlled with MouseKeys
. - (on systems with no center button, select both left and right mouse buttons)
- – Select the right mouse button to be controlled with MouseKeys

D.3.5.4 Adjustments:

— On/Off - MouseKeys (default is Off),

— On/Off - keyboard activation & deactivation of MouseKeys (default is On),

— setting for the top pointer speed,

— setting for the rate of acceleration (starting at very slow – 1 s for noticeable speed increase),

— On/Off - “MouseKeys when Num Lock On” (allows the user to chose which other keypad mode they want to use with MouseKeys) (default is On),

— On/Off - “Show MouseKeys Status on Screen” (optional) (default is On)

D.3.6 RepeatKeys

RepeatKeys is designed to allow use of computers by people who cannot move quickly enough when pressing keys to keep them from auto-repeating. The facility to adjust repeat onset, repeat rate and to turn auto-repeat off are usually included as part of most keyboard control panels. If these functions are not included, RepeatKeys provides them. RepeatKeys also ensures that the repeat delay and repeat interval can be set long enough for users who do not have quick response (if the maximum value for either of the regular key repeat settings are not long enough).

D.3.6.1 Operation:

These settings affect the auto-repeat function when keys are held down.

D.3.6.2 Adjustments:

— Key repeat On/Off,

— setting for repeat onset delay (maximum value of at least 2 s),

— setting for repeat interval (maximum value of at least 2 s).

D.3.7 ToggleKeys

ToggleKeys is a feature for users who cannot see the visual keyboard status indicators for locking (toggle) keys such as CapsLock, ScrollLock, NumLock, etc. ToggleKeys provides an auditory signal, such as a high beep, to alert the user that a toggle key such as the CapsLock has been locked, and a separate signal, such as a low beep, to alert the user that a toggle key has been unlocked.

D.3.7.1 Turn feature On/Off:

From the control panel

D.3.7.2 Operation:

Pressing any toggle key causes a tone to be sounded, a high tone indicating the key is now locked, and a low tone indicating the key is now unlocked.

D.3.7.3 Adjustments:

On/Off - ToggleKeys (default is Off)

D.3.8 SoundSentry

SoundSentry is a feature for individuals who cannot hear system sounds (due to hearing impairment, a noisy environment, or an environment where sound is not allowed such as a library or classroom). SoundSentry provides a visual signal (e.g. screen flash, caption bar flash, etc.) to visually indicate when the computer is generating a sound. SoundSentry works by monitoring the system sound hardware and providing a user selectable indication whenever sound activity is detected. Note that this feature usually cannot discriminate between different sounds, identify the sources of sounds, or provide a useful alternative for speech output or information encoded in sounds. Applications should support the ShowSounds feature (described below) to provide the user with a useful alternative to information conveyed using sound. SoundSentry is just a system-level fallback for applications that do not support ShowSounds.

D.3.8.1 Turn feature On/Off:

From the control panel.

D.3.8.2 Operation:

When SoundSentry is On all sounds cause the user-selected indicator to be activated.

D.3.8.3 Adjustments:

— On/Off - SoundSentry (default is Off),

— setting for the type of visual feedback (common ones are flash of on-screen icon, flash of full screen, flash of foreground window frame, flash of desktop).

D.3.9 ShowSounds

ShowSounds is a feature for users who cannot clearly hear speech or cannot distinguish between sounds from a computer due to hearing impairment, a noisy environment, or an environment where sound is not allowed such as a library or classroom. ShowSounds is a user configurable system flag that is readable by application software and is intended to inform ShowSounds-aware applications that all information conveyed audibly should also be conveyed visually. (E.g. captions should be shown for recorded or synthesized speech, and a message or icon should be displayed when a sound is used to indicate that new mail has arrived.

NOTE Captions should not be provided for speech output where the speech is reading information that is already visually presented on the screen (e.g. screen readers, etc).

D.3.9.1 Turn ShowSounds system flag On/Off:

From the control panel.

D.3.9.2 Adjustments:

True/False for ShowSounds flag (default is False).

D.3.10 Time Out (for all access features)

Time Out allows the access features to automatically turn off after an adjustable time when no keyboard or mouse activity occurs. Time Out is intended to be used on public or shared computers, such as those in libraries, bookstores, etc., where a user might leave the computer with an access feature turned On, thus potentially confusing the next user or leading people to think the computer was broken.

D.3.10.1 Turn feature On/Off:

From the control panel

D.3.10.2 Adjustments:

— On/Off - Time Out feature (default is Off),

— Setting for period of time of inactivity before access features are disabled (maximum of at least 30 min, default is 10 min).

D.3.11 SerialKeys

NOTE: SerialKeys was designed for users who are unable to use traditional keyboards and mice and must use a special communication aid or computer access aid to act as their keyboard and mouse. This functionality however is now

met by USB. Consequently SerialKeys is being retired. It is therefore not further specified or discussed in this document except the following brief description.

SerialKeys allows users to connect an external hardware device to the computer's serial port and send ASCII characters to the serial port which instruct SerialKeys to simulate standard keyboard and mouse events. To applications, keyboard and mouse events simulated by SerialKeys should be indistinguishable from events generated by the physical keyboard and mouse. For more information on SerialKeys including technical specifications for ASCII/Unicode strings supported by SerialKeys, see **http://trace.wisc.edu** and search for "GIDEI".

Annex E
(informative)

Accessibility and Usability

E.1 General

The purpose of this annex is to describe in more detail the basis of the definition of accessibility used in this standard and discuss the implications of using this definition. The approach adopted in this standard is that accessibility should be based on design targeting the widest possible range of user capabilities. Accessibility relates both to the process of design, in that it is a goal of design, and also to the product of the design process, in that it provides a basis for measuring the extent to which the product can be used.

In this part of ISO 9241, addressing software of interactive systems, accessibility describes the most important aspects of the users' ability to accomplish tasks. Accessibility is not a single or multi-dimensional measure for characterising the attributes of an interactive system (typically seen as a product or as a delivered service) but is a concept that relates to the interaction between the user and product or service, expressed in terms of the achievement of task goals. This means that producers, organisational users, end-users have to take responsibility for defining the purposes of the interactive system in question (mostly expressed in terms of tasks or groups of tasks, and their associated goals) and have to agree on the design objectives expressed in terms of accessibility. In some cases there will be legislative requirements, which provide the framework within which both task goals and accessibility targets are set.

E.2 Definition of accessibility

The definition of accessibility builds upon the definition of usability in ISO 9241-11:1998 which is "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". Usability is associated not with the product in itself, but with the interaction between the user and the product. The emphasis on the quality of the interaction is regarded as also being fundamental to the goal of increasing the levels of accessibility achieved in software products. The major shift in emphasis between the two concepts is to address the issue of the diversity of capabilities within potential user populations, i.e. the specified users.

It is clear that:

— The "specified users" in ISO 9241-11 include people with very wide ranging abilities and, very probably, some "people with disabilities". "Specified users" and "people with disabilities" are not separate groups. There may be some situations where the nature of the role limits the range of capabilities that can be accepted in the people who can undertake it, for example test pilots are selected on the basis of very specific criteria of capability. However, the general case is that user groups are highly diverse and as the use of interactive software systems becomes ubiquitous the variation in user characteristics increases.

— If users with disabilities, included in the set of specified users, cannot operate a product with the levels of effectiveness, efficiency and satisfaction that have been established as the targets for design, then the product has low levels of usability for those users and the level of accessibility achieved in relation to the specified user group will be also be low. This cannot be achieved simply by the adaptation of user interfaces for the disabled, but involves the systematic integration of the guidance into the design. This includes limiting the need to use assistive devices within what is feasible, and facilitating the connection of assistive devices when necessary.

— A product that cannot be used to achieve task goals is never going to be effective, and therefore is neither usable nor accessible. To the extent that the level of effectiveness can be enhanced and increased levels of efficiency and satisfaction can be achieved, the usability of the product is increased.

The ability to achieve increased levels of effectiveness, efficiency and satisfaction by specified user groups including increasingly diverse user capabilities will lead to increased accessibility. However, neither usability nor accessibility are absolute concepts since they have to be understood in relation to the context of use and the context of design.

Thus the concepts of accessibility and usability overlap and are interdependent, which has led to the definition combining usability and accessibility. For the purpose of this international standard, accessibility is defined as, *"the usability of a product, service, environment or facility by people with the widest range of capabilities".* This definition is consistent with the concept of Design for All, normally described as design of products, services, etc. to be used by people with the widest range of capabilities.

The conclusion to be drawn from this approach is that no single level of accessibility can be identified as the goal for all situations, since the aim will be to achieve higher levels of accessibility in the particular design situation, to the greatest extent possible.

E.3 Measurability of usability and accessibility

The definition of usability in ISO 9241-11 stresses the potential to achieve varying levels of usability and to obtain objective measures of these differences. One can measure the *effectiveness, efficiency and satisfaction* when specified users use a product to achieve specified goals in a specified context of use. In practice it is necessary to provide operational measures that are specific to the given design situation. The relative importance of the operational measures of each of the factors will vary from situation to situation. While the ability to be effective in relation to task goals is fundamental, the relative importance of achieving high levels of effectiveness may be less critical in some design contexts than the operational measures of other variables. For example, it may be critical to ensure that a consumer product enables the user to achieve the goals that the basic functionality supports, but the provision of additional functionality that can be used effectively, may not matter as much as ensuring that the product is aesthetically pleasing, and is measured as having high levels of satisfaction with ownership. For a given design situation the operational measures will provide the criteria against which the usability of the outcome is to be measured, and their relative importance will should be identified in the design requirements.

This means that if one wishes to compare usability one may only vary one entity at the time, while applying the same operational measures.

Usability (f operational measures of effectiveness, efficiency, satisfaction) = f (users, product, tasks/goals, context of use)

For example if you want to compare the usability of two different products you must test them with specified users who have the same characteristics, performing the same tasks in the same context of use.

From this it can be inferred that accessibility as defined in this standard is also measurable. However for higher levels of accessibility to be claimed it will be necessary to demonstrate that the levels of effectiveness, efficiency and satisfaction, set as the criteria for design, are achievable by users with a wider range of capabilities, i.e. the characteristics of the users included in the specified user group are widened so that people with a wider range of capabilities achieve equivalent levels of usability to those experienced in a group with a narrower range of capabilities.

E.4 Relationship with design guidance

Notwithstanding the importance of addressing accessibility in terms of the outcome for the users, it is possible to establish requirements and recommendations for software product attributes that are generally applicable to the creation of higher levels of accessibility. Frequently this is expressed in terms of how the software needs to behave in order to support differing user characteristics, and is thus dependent on this aspect of the context of use. However in some cases the guidance is more prescriptive in terms of the solution to be implemented, and in such cases this is based on established evidence that these solutions lead, in a general and wide-ranging way, to a greater diversity of user capabilities being accommodated. In some case guidance on an appropriate solution to adopt is specifically related to the context of use that has been identified, and thus

requires understanding of the context of use as a basis for application. The requirements and recommendations provided in this standard thus provide support for the development of solutions that will lead to increases in accessibility, taking account of the context of use.

Annex F
(informative)

Overview of the ISO 9241 series

This annex presents an overview of ISO 9241: its structure, subject areas and the current status of both published and projected parts, at the time of publication of this part of ISO 9241.

Part no.	Subject/title	Current status
1	General introduction	International Standard (intended to be replaced by ISO TR 9241-1 and ISO 9241-130)
2	Guidance on task requirements	International Standard
3	Visual display requirements	International Standard (intended to be replaced by the ISO 9241-300 subseries)
4	Keyboard requirements	International Standard (intended to be replaced by ISO 9241-400 subseries)
5	Workstation layout and postural requirements	International Standard (intended to be replaced by ISO 9241-500)
6	Guidance on the work environment	International Standard (intended to be replaced by ISO 9241-600)
7	Requirements for display with reflections	International Standard (intended to be replaced by the ISO 9241-300 subseries)
8	Requirements for displayed colours	International Standard (intended to be replaced by the ISO 9241-300 subseries)
9	Requirements for non-keyboard input devices	International Standard (intended to be replaced by the ISO 9241-400 subseries)
10	Dialogue principles	International Standard (intended to be replaced by ISO 9241-110)
11	Guidance on usability	International Standard
12	Presentation of information	International Standard (intended to be replaced by ISO 9241-111 and ISO 9241:141)
13	User guidance	International Standard (intended to be replaced by ISO 9241-121)
14	Menu dialogues	International Standard (intended to be replaced by ISO 9241-131)
15	Command dialogues	International Standard (intended to be replaced by ISO 9241-132)

Part no.	Subject/title	Current status
16	Direct-manipulation dialogues	International Standard (intended to be replaced by ISO 9241-133)
17	Form filling dialogues	International Standard (intended to be replaced by ISO 9241-134)
20	Accessibility guidelines for information communication equipment and services	Under preparation
Software Ergonomics		
100	Introduction to software ergonomics	Planned as ISO TR 9241-100
Guiding principles		
110	Dialog principles	To be published
111	Presentation principles	Planned to partially revise and replace ISO 9241-12
112	Multimedia principles	Planned to revise and replace ISO 14915-1
113	GUI and controls principles	Planned
User support (independent of dialogue techniques)		
121	User guidance	Planned to revise and replace ISO 9241-13:2003
122	Dialogue navigation	Planned to partially revise and replace ISO 14915-2, together with ISO 9241-142
Dialogue techniques (independent of technical implementation)		
130	Selection and combination of dialogue techniques	Planned to incorporate and replace ISO 9241-1:1997/Amd 1:2001
131	Menu dialogues	Planned
132	Command dialogues	Planned
133	Direct manipulation dialogues	Planned
134	Form filling dialogues	Planned
135	Natural language dialogues	Planned
(Technology-dependent) interface components and technologies		
141	Window interfaces	Planned
142	Controls	Planned to partially revise and replace ISO 14915-2, together with ISO 9241-122
143	Media selection & combination	Planned to revise and replace ISO 14915-3
Application-area specific		
151	Guidance on World Wide Web software user interfaces	Under preparation

Part no.	Subject/title	Current status
152	Virtual reality	Planned
Accessibility		
171	Guidance on software accessibility	Under preparation
Human centred design		
200	Introduction to human-system interaction processes	Planned
201	Business case for human centred design	Planned
202	Terminology for human-centred design	Planned
210	Guidance on Human centred design	Planned
211	Guidance for HCD project managers	Planned to revise and replace ISO 13407
Process		
221	Specification for the process assessment of human-system issues	Planned to revise and replace ISO/PAS 18152
222	Human-centred lifecycle process descriptions	Planned to revise and replace ISO/TR 18529
Methods		
231	Usability methods supporting human-centred design	Planned to revise and replace ISO/TR 16982
Evaluation		
241	Templates for evaluation	Planned
Ergonomic requirements and measurement techniques for electronic visual displays		
300	Introduction to electronic visual display requirements	Under preparation
302	Terminology for electronic visual displays	Under preparation
303	Requirements for electronic visual displays	Under preparation
304	User performance test methods for electronic visual displays	Under preparation
305	Optical laboratory test methods for electronic visual displays	Under preparation
306	Field assessment methods for electronic visual displays	Under preparation
307	Analysis and compliance test methods for electronic visual displays	Under preparation
Physical input devices		
400	Principles and requirements for physical input devices	To be published
410	Design criteria for physical input devices	Under preparation
411	Laboratory test and evaluation methods for the design of physical input devices	Planned
420	Selection procedures for physical input devices	Under preparation

Part no.	Subject/title	Current status
421	Workplace test and evaluation methods for the use of physical input devices	Planned as ISO TR 9241-421
Workstation		
500	Workstation layout and postural requirements	Planned to revise and replace ISO 9241-5
Work environment		
600	Guidance on the work environment	Planned to revise and replace ISO 9241-6
Control centres		
710	Introduction to ergonomic design of control centres	Planned
711	Principles for the design of control centres	Planned to revise and replace ISO 11064-1
712	Principles for the arrangement of control suites	Planned to revise and replace ISO 11064-2
713	Control room layout	Planned to revise and replace ISO 11064-3
714	Layout and dimensions of control centre workstations	Planned to revise and replace ISO 11064-4
715	Control centre displays and controls	Planned to revise and replace ISO 11064-5
716	Control room environmental requirements	Planned to revise and replace ISO 11064-6
717	Principles for the evaluation of control centres	Planned to revise and replace ISO 11064-7
Tactile and haptic interactions		
900	Introduction to tactile and haptic interactions	Planned
910	Framework for tactile and haptic interactions	Planned
920	Guidance on tactile and haptic interactions	Under preparation
930	Haptic and tactile interactions in multimodal environments	Planned
940	Evaluation of tactile and haptic Interactions	Planned
971	Haptic and tactile interfaces to publicly available devices	Planned

Bibliography

[1] ISO 13406-2:2001, *Ergonomic requirements for work with visual displays based on flat panels — Part 2: Ergonomic requirements for flat panel displays*

[2] ISO/DIS 9241-302:2006*, Ergonomics of human-system interaction — Part 302: Terminology for electronic visual displays*

[3] ISO/IEC 11581-1:2000, *Information technology — User system interfaces and symbols — Icon symbols and functions — Part 1: Icons — General*

[4] BERGMAN, E., JOHNSON, E. (1995). *Towards Accessible Human-Computer Interaction*, Advances in HCI, Volume 5, Ablex Publishing Corporation

[5] BLATTNER, M.M., GLINERT, E.P., JORGE, J.A. and ORMSBY, G.R. (1992). *Metawidgets: Towards a theory of multimodal interface design*. Proceedings: COMPASAC 92, IEEE Press, pp. 115-120

[6] BROWN, C. (1989). *Computer Access in Higher Education for Students with Disabilities*, 2nd Edition. George Lithograph Company, San Francisco

[7] BROWN, C. (1992). *Assistive Technology Computers and Persons with Disabilities*, Communications of the ACM, 35 (5), pp. 36-45

[8] CARTER, J., AND FOURNEY, D. (2004). *Using a Universal Access Reference Model to identify further guidance that belongs in ISO 16071*. *Universal Access in the Information Society*, 3 (1), p. 17–29. http://www.springerlink.com/link.asp?id=wdqpdu5pj0kb4q6b

[9] CASALI, S.P. and WILLIGES, R.C. (1990). *Data Bases of Accommodative Aids for Computer Users with Disabilities*, Human Factors, **32**(4), pp. 407-422

[10] CHISHOLM, W., VANDERHEIDEN, G. AND JACOBS, I. (eds), 1999 Web Content Accessibility Guidelines 1.0 W3C, Cambridge, MA, USA url: www.w3.org/TR/WAI-WEBCONTENT/

[11] CHURCH, G. and GLENNEN, S. (1992). *The Handbook of Assistive Technology*, Singular Publishing Group, Inc., San Diego

[12] EDWARDS, W.K., MYNATT, E.D., RODRIGUEZ, T. (1993). *The Mercator Project: A Nonvisual Interface to the X Window System*. The X Resource. O'Reilly and Associates, Inc.

[13] EDWARDS, A., EDWARDS, A. and MYNATT, E. (1993). *Enabling Technology for Users with Special Needs*, InterCHI '93 Tutorial, 1993

[14] ELKIND, J. (1990). *The Incidence of Disabilities in the United States*, Human Factors, 32 (4), pp. 397-405

[15] EMERSON, M., JAMESON, D., PIKE, G., SCHWERDTFEGER, R. and THATCHER, J. (1992). *Screen Reader/PM*. IBM Thomas J. Watson Research Center, Yorktown Heights, NY

[16] GLINERT, E.P. and YORK, B.W. (1992). *Computers and People with Disabilities,* Communications of the ACM, 35 (5), pp. 32-35

[17] GRIFFITH, D. (1990). *Computer Access for Persons who are Blind or Visually Impaired: Human Factors Issues*. Human Factors, 32 (4), 1990, pp. 467-475

[18] IBM Technical Report (1988). *Computer-Based Technology for Individuals with Physical Disabilities: A Strategy for Alternate Access System Developers*

[19] KAPLAN, D., DEWITT, J., STEYAERT, M. (1992). *Telecommunications and Persons with Disabilities: Laying the Foundation World Institute on Disability*

[20] KUHME, T. *A User-Centred Approach to Adaptive Interfaces*. (1993). Proceedings of the 1993 International Workshop on Intelligent User Interfaces. Orlando, FL., New York: ACM Press, pp. 243-246

[21] LAZZARO, Joseph J. (1993). *Adaptive Technologies for Learning and Work Environments*. American Library Association, Chicago and London

[22] *Macintosh Human Interface Guidelines*. (1992). Addison-Wesley

[23] *Managing Information Resources for Accessibility*, U.S. General Services Administration Information Resources Management Service. (1991). Clearinghouse on Computer Accommodation

[24] MCCORMICK, John A. (1994). *Computers and the American's with Disabilities Act: A Manager's Guide*. Windcrest

[25] MCMILLAN, W.W. (1992). *Computing for Users with Special Needs and Models of Computer Human Interaction*. Conference on Human Factors in Computing Systems, CHI '92, pp. 143-148. Addison Wesley

[26] MICROSOFT CORPORATION. *The Windows Interface Guidelines for Software Design*. (1995). Microsoft Press.

[27] MICROSOFT CORPORATION. (2001). *Microsoft Active Accessibility Version 2.0*. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/msaa/msaastart_9w2t.asp

[28] MYNATT, E. (1994). *Auditory Presentation of Graphical User Interfaces*, in Kramer, G.(ed), Auditory Display: Sonification, Audification and Auditory Interfaces, Santa Fe. Addison Wesley: Reading MA

[29] NEWELL, A.F. and CAIRNS, A. (1993). *Designing for Extraordinary Users*. Ergonomics in Design

[30] NIELSEN, J. (1993). *Usability Engineering*. Academic Press, Inc., San Diego

[31] NSI T1.232:1993, *Operations, Administration, Maintenance, and Provisioning (OAM&P) – G Interface Specification for Use with the Telecommunications Management Network (TMN)*

[32] OZCAN, O. *"Feel-in-Touch: Imagination through Vibration"* (2004). Leonardo. MIT Press, Volume 37, No 4, pp 325-330.

[33] PERRITT Jr., H.H. (1991). *Americans with Disabilities Act Handbook*, 2nd Edition. John Wiley and Sons, Inc., New York

[34] *Resource Guide for Accessible Design of Consumer Electronics* (1996). EIA/EIF

[35] SAUTER, S.L., SCHLEIFER, L.M. and KNUTSON, S.J. (1991). *Work Posture, Workstation Design, and Musculoskeletal Discomfort in a VDT Data Entry Task.* Human Factors, 33 (2), pp. 407-422

[36] SCHMANDT, C. (1993). *Voice Communications with Computers. Conversational Systems*. Van Nostrand Reinhold, New York, 1993

[37] SUN MICROSYSTEMS, INC. "*JAVA ACCESSIBILITY. Overview of the Java Accessibility Features. Version 1.3*". (1999).. http://java.sun.com/products/jfc/jaccess-1.3/doc/guide.html

[38] THATCHER, J., BURKS, M., SWIERENGA, S., WADDELL, C., REGAN, B., BOHMAN, P., HENRY, S., and URBAN, M. (2002). *Constructing Accessible Web Sites*. Glasshaus, Birmingham, U.K.

[39] THORÉN, C. (ed.) (1998). *Nordic Guidelines for Computer Accessibility*. Second Edition. Vällingby, Sweden: Nordic Committee on Disability

[40] VANDERHEIDEN, G.C. (1983). *Curbcuts and Computers: Providing Access to Computers and Information Systems for Disabled Individuals*. Keynote Speech at the Indiana Governor's Conference on the Handicapped

[41] VANDERHEIDEN, G.C. (1988). *Considerations in the design of Computers and Operating Systems to increase their accessibility to People with Disabilities*, Version 4.2, Trace Research & Development Center

[42] VANDERHEIDEN, G.C. (1990). *Thirty-Something Million: Should they be Exceptions?* Human Factors, 32 (4), p. 383-396

[43] VANDERHEIDEN, G.C. (1991). *Accessible Design of Consumer Products: Working Draft 1.6*. Trace Research and Development Center, Madison, Wisconsin

[44] VANDERHEIDEN, G.C. (1992). *Making Software more Accessible for People with Disabilities: Release 1.2*. Trace Research and Development Center, Madison, Wisconsin

[45] VANDERHEIDEN, G.C. (1992). *A Standard Approach for Full Visual Annotation of Auditorily Presented Information for Users, Including Those Who are Deaf: Show sounds*. Trace Research & Development Center

[46] VANDERHEIDEN, G.C. (1994). *Application Software Design Guidelines: Increasing the Accessibility of Application Software to People with Disabilities and Older Users, Version 1.1*. Trace Research & Development Center

[47] WAI ACCESSIBILITY GUIDELINES: *User Agent Accessibility Guidelines*. (2000). http://www.w3c.org/wai

[48] WAI ACCESSIBILITY GUIDELINES: *Web Content Accessibility Guidelines 1.0* (1999). http://www.w3.org/TR/WCAG10/

[49] WAI ACCESSIBILITY GUIDELINES: *Web Content Accessibility Guidelines 2.0* (2005). http://www.w3.org/TR/WCAG20/

[50] WALKER, W.D., NOVAK, M.E., TUMBLIN, H.R., VANDERHEIDEN, G.C. (1993). *Making the X Window System Accessible to People with Disabilities*. Proceedings: 7th Annual X Technical Conference. O'Reilly & Associates

[51] WORLD HEALTH ORGANIZATION, (2002) *Towards a common language for functioning, disability and health: ICF*, WHO/EIP/CAS/01.3, World Health Organisation, Geneva, http://www3.who.int/icf/