TECHNICAL SPECIFICATION

ISO/TS 16071

First edition
2003-02-01

Ergonomics of human-system interaction — Guidance on accessibility for human-computer interfaces

Ergonomie de l'interaction homme/système — Guidage relatif à l'accessibilité aux interfaces homme/ordinateur

Reference number
ISO/TS 16071:2003(E)

Contents

Page

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.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50 % of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed after three years in order to decide whether it will be confirmed for a further three years, revised to become an International Standard, or withdrawn. If the ISO/PAS or ISO/TS is confirmed, it is reviewed again after a further three years, at which time it must either be transformed into an International Standard or be withdrawn.

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/TS 16071 was prepared by Technical Committee ISO/TC 159, *Ergonomics*, Subcommittee SC 4, *Ergonomics of human-system interaction*.

Introduction

The purpose of this Technical Specification is to provide guidance to developers on designing human-system interfaces which can be used with 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

— task-oriented design of user interfaces,

— individualization (see ISO 9241-10),

— the use of human-centred design principles (see ISO 13407),

— individualized user instruction and training, and

— enabling the efficient use of assistive technologies.

The focus of this Technical Specification is the development of human-system interfaces to systems and products that are intended for use by the widest range of people with special needs. An important part of a human-centred design process for accessible systems is to develop human-system interfaces to meet accessibility goals that can be evaluated for a specific user or user category in a specified context of use.

This Technical Specification is based mainly on the prevalent knowledge of individuals who have sensory and/or motor impairments. However, accessibility is an attribute that affects a large variety of capabilities and preferences of human beings. These different capabilities may be the result of age, disease and/or disabilities. Therefore, accessibility addresses a widely defined group of users including

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

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

— 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 certain situations, such as a person who works in a noisy environment or has both hands occupied by other work.

It should be emphasized that having a disability should be regarded as a natural element of human life. Everyone can expect, during some period of life, to be affected by circumstances that make it difficult to access and use systems, products and services.

This Technical Specification recognizes that some users will always need assistive devices in order to use a system. Therefore, this Technical Specification includes, in the concept of accessibility, the capability of a system to connect and interact successfully with assistive technologies.

Guidance is provided for system design, appearance and behaviour. The guidance will allow software to be used by as broad an audience as possible. In addition, guidance will be provided on designing software that integrates as effectively as possible with common assistive technologies (e.g. speech synthesizers, Braille input and output devices) when they are available. Incorporating accessibility features early in the design process is relatively inexpensive compared to the cost of modifying products to make them accessible.

This Technical Specification addresses the increasing need to consider social and legislative demands to ensure accessibility by removing barriers that prevent people with special requirements from participating in life activities including the use of environment, services, products and information. Designing software-user interfaces for accessibility increases the number of people who can use systems by taking into account the varying physical and sensory capabilities of user populations. Designing for accessibility benefits disabled users by making software easier for them to use, or making the difference in whether they are able to use the software at all.

Many accessibility features also benefit users who do not have a disability, by enhancing usability and providing additional customization possibilities. They may also help to overcome a temporary defect (e.g. a broken arm or hand). They benefit designers and suppliers by expanding the number of potential users (and thus sales for their product) and often by making the product compliant with legal requirements for accessibility. They benefit companies buying software by expanding the number of employees who may use the software.

It is important to note that accessibility may be provided by a combination of both hardware and 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 screen 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 that replace mice and Braille output devices that replace a video display. There are many other examples not listed here. When users provide 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 may have to provide “hooks” or other features to allow software to operate effectively with add-on assistive software and hardware as recommended in these guidelines. If systems do not provide support for assistive technologies, the probability increases that users will encounter problems with compatibility, performance and usability. At the same time, if software applications do not use system-provided mechanisms (such as customization for colour, font, and audio, or system routines for keyboard navigation and text input), then users may find their access blocked.

This Technical Specification 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;

— buyers, who will reference this Technical Specification during product procurement;

— evaluators, who are responsible for ensuring that products meet the recommendations of this Technical Specification.

The ultimate beneficiary of this Technical Specification will be the end user of the software. Although it is unlikely that the end-users will read this Technical Specification, its application by designers, developers, buyers and evaluators should provide user interfaces that are more accessible. These guidelines concern the development of software for user interfaces. However, those involved in designing the hardware aspects of user interfaces may also find them useful.

Ergonomics of human-system interaction — Guidance on accessibility for human-computer interfaces

1 Scope

This Technical Specification provides guidance on the design of accessible (work, home, education) software. It covers issues associated with designing accessible software for people with the widest range of visual, hearing, motor and cognitive abilities, including those who are elderly and temporarily disabled. This Technical Specification addresses software considerations for accessibility that complement general design for usability covered by ISO 9241-10 to ISO 9241-17 and ISO 13407.

This Technical Specification addresses the accessibility of interactive systems. It addresses a wide range of solutions, including office applications, web pages and multimedia. It does not provide recommendations for the design of hardware.

This Technical Specification promotes increased usability of systems in combination with assistive technologies, when they are required. It does not cover the behaviour or requirements of assistive technologies themselves (including assistive software).

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 9241-10:1996, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 10: Dialogue principles*

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 9241-14:1997, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 14: Menu dialogues*

ISO 9241-15:1997, *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 9241-17:1998, *Ergonomic requirements for office work with visual display terminals (VDTs) — Part 17: Form filling dialogues*

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

3 Terms and definitions

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

3.1
accelerator keys
key combinations (sometimes called “shortcut keys”) which invoke a menu option without displaying the menu on which the option appears or intermediate menus

[ISO 9241-14:1997, definition 3.1]

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

NOTE Although “accessibility” typically addresses users who have a disability, the concept is not limited to disability issues.

3.3
activation
initiation of an action associated with a selected object

3.4
assistive technologies
hardware or software products used by people with disabilities to accomplish their tasks

EXAMPLES Braille displays, screen readers, screen magnification software, and eye tracking devices.

3.5
bounce keys
feature that allows users to set a delay after a keystroke during which an additional keystroke will not be accepted if it is identical to the previous keystroke

NOTE This feature is often implemented in the system software.

3.6
chorded key-press
keyboard or pointer-button presses where more than one button is held down simultaneously to invoke an action

3.7
contrast
⟨in a perceptual sense⟩ assessment of the difference in appearance of two or more parts of a field seen simultaneously or successively (hence: brightness contrast, lightness contrast, colour contrast, etc.)

NOTE Adapted from IEC 60050 (845-02-47):1987.

[ISO 13406-2:2001]

3.8
colour palette
⟨in computer graphics⟩ a fixed set or range of available colours that can be selected

3.9
cursor
visual indication of the focus for alphanumeric input

[ISO 9241-12:1998, definition 3.4]

NOTE See also **focus cursor** (3.12) and **text cursor** (3.35). Contrast with **pointer** (3.28).

3.10
disability
impairment that interferes with the customary manner in which a task is performed or that requires accommodation in order to perform a task

NOTE Note that the legal definitions of a disability vary from country to country, and may differ from the definition stated here.

3.11
explicit focus
setting in which the input focus is transferred in response to an explicit user action, not simply when the pointer passes or pauses over an object

EXAMPLE 1 Clicking when the pointer is over an object.

EXAMPLE 2 Pressing the alt-tab key combination.

NOTE Windows may have a limited area, such as a title bar or frame, in which explicit focus must be assigned.

3.12
focus cursor
indicator showing which user interface object within a window has focus

EXAMPLE A box or highlighted area around a text field, button, list, or menu option.

NOTE 1 Also called “location cursor”. See also input focus, text cursor, and focus indicator.

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

3.13
focus indicator
indicator that shows which window or pane has focus

EXAMPLE 1 A change in border colour, so that the window with focus has a border of one colour and all other windows have a border of a single, noticeably different colour.

EXAMPLE 2 A change in visible detail, so that the window with focus shows the full details and shading of the title bar, scroll bars, etc., and all other windows show only the outlines.

NOTE See also **input focus** (3.17), **text cursor** (3.34) and **focus cursor** (3.12).

3.14
impairment
any deficit in psychological, physiological or anatomical structure or function

NOTE An impairment is not a disability if it does not interfere with task performance. See also **disability** (3.10).

3.15
implicit designator
mnemonic
menu mnemonic
portion of an option name or control label used for a keyboard

3.16
implicit focus
setting in which input focus is transferred when the pointer passes or pauses over a window or object

NOTE 1 Contrast with **explicit focus** (3.11). See also **keyboard focus** (3.20).

NOTE 2 Keyboard navigation provides implicit focus by giving focus to whatever object is currently indicated by the focus cursor.

3.17
input focus
current assignments of the input from an input device to a user-interface object

EXAMPLES Pointer focus and keyboard focus.

3.18
keyboard
hardware device (or logical equivalent) consisting of a number of mechanical buttons (keys) that the user presses to input characters to a system

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

3.19
keyboard equivalents
keys or key combinations that provide keyboard access to functions usually activated by a pointing device

3.20
keyboard focus
current assignment of the input from the keyboard or equivalent to a user interface object

NOTE For a window, focus is indicated by a focus indicator; for an individual object, focus is indicated by a focus cursor.

3.21
latch
mode in which any modifier key remains logically pressed (active) in combination with a single subsequent non-modifier keypress

NOTE Contrast with **lock** (3.22).

3.22
lock
persistent mode in which any modifier key remains logically pressed (active) in combination with any number of subsequent key-presses until lock mode is turned off

NOTE Contrast with **latch** (3.21).

3.23
mnemonic code
code conveying information that is meaningful to the user and has some association with the words it represents

NOTE Mnemonic codes frequently consist of alphanumeric characters, making them easier to learn and recall. Many mnemonic codes are abbreviations.

[ISO 9241-12:1998, definition 3.2.1]

3.24
modifier key
keyboard key that changes the action or effect of another key or of the pointing device

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

EXAMPLE 2 The control or command key transforms the keyboard keys from text input into commands.

3.25
mouse keys
system software feature providing keyboard control of pointer movement and mouse button functions

3.26
object
entity which is presented to the user during the dialogue

NOTE Both entities relevant for the task (such as a letter, a sales order, electronic parts, a wiring diagram) and entities of the user interface (such as an icon, a window, a push button) are regarded as objects. Different object types are text objects, graphical objects or control objects. It may be possible for the user to directly manipulate some of these objects.

[ISO 9241-16:1999, definition 3.13]

3.27
pointer
graphical symbol that is moved on the screen according to operations with a pointing device

[ISO 9241-16:1999, definition 3.15]

NOTE Although the pointer is sometimes called a “pointing cursor”, this Technical Specification uses the word cursor only for a keyboard input focus or location. See also **focus cursor** (3.12) and **text cursor** (3.34).

3.28
pointer focus
current assignment of the input from the pointing device to a window

NOTE The window with pointer focus usually has some distinguishing characteristic, such as a highlighted border and/or title bar.

3.29
repeat keys
system software feature that enables users to adjust the delay prior to the onset of key repeat, and the rate at which keys repeat, allowing users who have limited coordination to release keys

3.30
screen reader
assistive technology in combination with information available via the operating system that allows users to navigate through windows, determine the state of controls, and read text through Braille or text-to-speech conversion

3.31
show sounds
system software feature that provides a user-settable software flag giving a visual alternative for sounds

EXAMPLES flashes to accompany beeps, closed captions alongside speech.

3.32
slow keys
system software feature that enables users to adjust the delay prior to key-press acceptance

NOTE This prevents keys which are accidentally pressed from being accepted, so that only purposefully pressed keys are accepted.

3.33
sticky keys
system software feature that enables users to latch or lock modifier keys alone or in combination

NOTE This feature is especially useful for users who have physical impairments that make it difficult or impossible to press and/or hold multiple keys simultaneously.

3.34
text cursor
visual indication of the current insertion point for text entry

NOTE Contrast with pointer and focus cursor.

3.35
toggle keys
system software feature that enables users to have the system produce a sound when the current state of a binary-state keyboard toggle-control changes

3.36
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, definition 3.1]

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 usability for people with the widest possible range of capabilities.

Accessibility can be improved by incorporating features and attributes known to benefit users with special requirements. In order to determine the achieved level of accessibility, it is necessary to measure the performance 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 Technical Specification 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 (effectiveness and efficiency) 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 degree of 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 How to use this Technical Specification

Readers of this Technical Specification need to understand what parts of the requirements for software accessibility they are responsible for implementing. For this reason, this Technical Specification has been categorized for two dimensions: accessibility impact category (see Table 1) and the part(s) of the system that should support the capabilities described (see Table 2).

Table 1 — Accessibility Impact

Impact category	Consequences if item in this category is NOT supported
Core	A substantial subset of users cannot accomplish most tasks without either additional assistance or extensive extra effort to compensate for the omission of the item. For these users, the system is unusable.
Primary	Most users can accomplish most tasks, but a substantial subset of users may not be enabled to perform some tasks, either due to lack of access to functionality or severe usability problems.
Secondary	Most users can accomplish most tasks, but a substantial subset of users may not be enabled to perform some tasks without a severe decrease in efficiency and satisfaction.

Table 2 — Implementation responsibility

Category	Description
OS (operating system)	Operating system, drivers, and associated software layers.
Application (productivity applications) (development tools) (web browsers), etc.	Software not considered as part of the operating system or its immediate layers. This includes “desktop” software bundled with an operating system, personal productivity applications, and other non-OS software.

Achieving accessibility requires support from operating systems and their layered components as well as applications. While application developers can do much to improve accessibility, applications alone cannot provide all of the input and output support that users require in every circumstance. Many users depend upon assistive software and hardware that inserts itself into the communication between the user and system.

Because of the availability of assistive technologies, every application does not have to provide every assistive solution (for example, each word processor and web browser need not implement a Braille output protocol or a voice input capability), but every application needs to support the use of assistive technologies for users with disabilities (see especially those guidelines marked “application” in Clause 7).

To the extent that operating systems provide built-in assistive technologies such as voice recognition, users with disabilities require less assistive technology. Just as applications need not provide a solution for every user who has a disability, neither does an operating system have to provide assistive solutions for all potential users. However, it is the intent of this Technical Specification that operating-system developers build key accessibility capabilities into every operating system and associated layers, and provide architectures in which operating systems, drivers, and related “middleware” support accessibility required by applications and assistive technologies (see guidelines marked “OS” in Clause 7).

Recommendations marked with “OS/application” in Clause 7 are the responsibility of the operating system as long as the application uses the operating-system controls and features. If the application takes control of relevant resources and features (i.e. by-passes the operating system) or implements features covered by the recommendations, it is the application’s responsibility to implement the recommendation.

6 Characteristics of users with special requirements

6.1 Variations in user characteristics

The user characteristics of people with any given impairment or disability vary significantly, just as in any other heterogeneous population. The descriptions contained here provide only an outline of the issues typically encountered by individuals with various disabilities, and do not constitute a comprehensive list. People may concurrently experience more than one of the disabilities outlined in 6.2 to 6.11. The needs of people who have such combinations of disabilities are covered in several cases by the overlap across guidelines.

6.2 Issues commonly encountered by users who are blind

Blind users have vision which is limited to the extent that they must use a non-visual interface (auditory and/or tactile) as their means of interaction with a system. The primary issue for blind users is how to obtain information provided by visual presentation, how to navigate among objects on screens, how to identify those objects, and how to control focus, navigation, and other functions via the keyboard.

Many people who are blind from birth learn Braille, and many who become blind later in life often rely on additional auditory methods to obtain information.

Many blind users interact with systems through “screen readers”, i.e. assistive software that can provide spoken or Braille information for windows, controls, menus, images, text and other information typically displayed visually on a screen.

Considerations for these users follow from the characteristics of interactions mediated by screen readers. To the extent that interactions depend on understanding a spatial metaphor for navigation or seeing a graphically represented object, blind users are more likely to encounter difficulties.

In addition, because many blind users read screens by means of synthesized speech output, they may find it difficult or impossible to attend to auditory outputs that occur while they are reading.

6.3 Issues commonly encountered by users who have low vision

Users who have low vision commonly have difficulty reading a standard screen display because of loss of visual acuity, colour-perception deficits, impaired contrast sensitivity, and/or loss of depth perception. Low-vision issues also include limited field of view (e.g. so-called tunnel vision), which provides only a subset of the information presented on a standard screen. People with vision deficiencies, even with the best possible correction, often have difficulties reading ordinary text.

People who have low vision use varying means of increasing the size, contrast and overall visibility of visual displays depending upon their visual needs. Common assistive technologies include the use of oversized monitors, large fonts, high contrast, and hardware or software magnification to enlarge portions of the display.

When interacting with systems, these users may not detect size coding, have difficulty with font discrimination, and encounter difficulties locating or tracking interface objects such as pointers, cursors, drop targets, hot spots, and direct manipulation handles.

Additionally, both blind users and users who have low vision experience difficulties when required to read very small displays, such as those on printers, copiers and ticket machines.

6.4 Issues commonly encountered by users who are deaf

A deaf user is anyone who does not perceive amplified speech even with a hearing aid. A common definition of “deaf” is the inability to hear sound below 90 dB. In addition to a general inability to detect auditory information, the issues commonly faced by deaf users may include the inability to produce speech recognizable by voice-input systems, and experience with a national language only as a second language (sign language often being the primary language for people who become deaf at an early age or who are born deaf).

If it is available, deaf users will typically use the “Show sounds” feature that notifies software to present audio information in visual form.

When interacting with systems, these users will encounter problems if important information is presented only in audio form. Many of these issues apply to any user in contexts where sound is masked by background noise (e.g. machine shop floor) or where sound is turned off or cannot be used (e.g. a library).

6.5 Issues commonly encountered by users who have a hearing loss

The issues commonly faced by users who have a hearing loss but who retain some functional hearing include the following:

— inability to discriminate frequency changes, decreased frequency range and dropout;

— difficulties localizing sounds;

— difficulty picking up sounds against background noise even with the use of a hearing aid.

Users who are deaf or hard of hearing may or may not use electronic hearing aids, depending on the nature and extent of the hearing impairment. If it is available in the operating system, they may use the "Show sounds" feature that notifies software to present audio-information in visual form.

When interacting with systems, these users may have trouble hearing sounds of certain frequencies, or of low volume. Sound individualization is essential for providing them with access.

6.6 Issues commonly encountered by users who have physical impairments

A person with a physical impairment is anyone whose motor functions are significantly limited. The issues commonly faced by users who have physical impairments often follow from physical limitations including poor co-ordination, weakness, difficulty reaching, and inability or difficulty moving a limb.

Users with physical impairments may or may not use assistive technologies, and the variety of hardware and software they employ is too large to describe in detail here. A few examples, however, include eye-tracking devices, on-screen keyboards, speech recognition, and alternative pointing devices.

Some users may have difficulty directly manipulating objects, using modifier keys or pointing devices, and performing actions that require precise movement or timing. Other users may have tremors that cause difficulty in moving to a target. The extreme variation in needs and capabilities among this user population means that individualization of input parameters and timing is extremely important for effective access.

6.7 Issues commonly encountered by users who have cognitive impairments

The issues commonly encountered by users who have cognitive disabilities involve difficulties receiving information, processing it, and communicating what they know. People with these impairments may have trouble learning new things, making generalizations 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 difficulties reading text that is presented in written form and difficulties in producing written text.

Reading difficulties can be supported by having the text highlighted and read out loud or by providing "easy-reading" versions of the texts. Users without reading difficulties also benefit from easy-reading versions of written text.

Providing a support of synthetic speech for what is to be written can aid writing difficulties.

6.8 Issues commonly encountered by users who are elderly

Elderly users may be progressively limited in their ability to use and access human-system interfaces due to the effects of visual, hearing, cognitive and motor impairments that, to varying degrees, come with increasing age. These may occur in multiple combinations.

Sometimes the awareness of diminishing capabilities is a concern for elderly users. Therefore, built-in product accessibility features may contribute to removing the stigma of special aids or modifications. Elderly persons do not want to have their age regarded as a disability.

6.9 Issues commonly encountered by users who have temporary disabilities

Temporary disabilities are often of a physical nature (e.g. a broken arm). These users seldom adopt efficient skills in learning to cope with their disability. It is therefore important to make the accessibility features for these disabilities easy to find and learn to master.

Temporary disabilities might also be caused by repetitive strain caused by poor ergonomics and intensive use of the system. It is then important that this injury can be relieved through support that can be provided to the user via the system.

6.10 Issues commonly encountered by users who have multiple disabilities

There are not just a few categories of disabilities, rather the range of different accessibility needs varies, just as the combinations of certain degrees of disabilities are a fact. For example, an individual with a cognitive impairment might also have low vision.

Several of the guidelines for addressing a specific disability might be contradictory for a person with multiple disabilities. For example, auditory output of written text is not a support for the deaf blind. It is therefore important that the support for these forms of multiple disabilities be individualized for the specific user and task.

6.11 Issues commonly encountered by users who experience a disabling environment

Disabling environments occur when specific features of the environment cause difficulties in perceiving signals from the system. Such situations include difficulties hearing signals from the system, for example, working in a noisy environment. These situations must be regarded as disabling the user to fulfil the task with the aid of the system. Although this may not be a software requirement, the immediate solution should be to improve the environment, and, in situations where this cannot be done, such as airports, provide redundant presentations of essential information.

7 Guidelines

7.1 Support for accessibility

Although it is not possible to make all systems accessible without add-on assistive technologies, these guidelines should help software designers develop software that is accessible when assistive technologies are not required, and provide the necessary interface information so that assistive software and devices can operate effectively and efficiently when used.

At the operating system level, support for accessibility should be provided. The system 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 ([29] in the Bibliography). For example, systems that provide built-in screen magnification eliminate the need for add-on magnification hardware or software, but if the necessary integration information is provided, users may attach a screen magnification program of their choice.

7.2 General guidelines

7.2.1 Input/output alternatives

7.2.1.1 Enable user input/output choice (core: OS/application)

The system should provide the necessary information to enable users to use as many input and output alternatives as possible ([8], [29] and [34] in the Bibliography).

EXAMPLE 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.

7.2.1.2 Enable switching of input/output alternatives (secondary: OS/application)

Users should be enabled to switch among input/output alternatives without requiring them to reconfigure or restart the system or applications.

NOTE This transparency aids non-impaired users and impaired users working on the same system.

EXAMPLE 1 A blind user uses his or her 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 object and speaks the word “Print” to print the contents, while another user presses a key sequence, and a third user chooses a menu item.

7.2.2 Enable user to perform the task effectively with any single input device (core: OS/application)

Users should be enabled to perform tasks effectively using only one input device, such as a pointing device ([34] in the Bibliography), voice or only a keyboard (or an alternative input device) (see ISO 9241-16:1999, subclauses 5.4.1 and 5.4.2).

EXAMPLE 1 Users enter text using an on-screen keyboard, and perform all other functions through menus, buttons and screen commands. A user with no movement capabilities in any limbs uses only a pointing device such as an eye-tracker or mouth-operated joystick.

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

EXAMPLE 3 All interface objects 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.

7.2.3 Provide user-preference profiles (secondary: OS/application)

Users should be enabled to easily create, save, edit and recall profiles of preferences, including input and output characteristics. To do this, it should not be necessary to restart the system or applications ([13] in the Bibliography).

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

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

EXAMPLE 2 An 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.

7.2.4 Enable user setting of timed responses (core: OS/application)

If a task requires users to make responses (e.g. press a button or type information) within a limited time in order for that response to be valid, the time range should be adjustable by the user, including the option to turn off all timing requirements ([31], [34] and [39] in the Bibliography).

7.2.5 Provide object descriptions (secondary: OS/application)

Where tasks require access to the visual content of objects beyond what a label provides, software should provide object descriptions that are meaningful to users stored as accessible text, whether those descriptions are visually presented or not ([34] in the Bibliography). Visual objects that are primarily decorative and contain little or no information need not be described.

NOTE Users who have low vision or are blind may use software that can present text descriptions to users who cannot view visually displayed objects. Descriptions exist independently from labels that name an object. Descriptions describe object contents in detail.

EXAMPLE 1 An image labelled "Construction Site" might have the following description: "A large building partly constructed sits in the middle of a muddy lot on the corner of two busy city streets. A crane is hoisting an enormous steel beam high into the air, while a dump truck empties gravel into a growing pile at the edge of a chain link fence."

EXAMPLE 2 A map image has a label "Map of Europe", with a description "A map depicts Western Europe, with a jagged line across Spain indicating where the glacial advance stopped in the last ice age."

EXAMPLE 3 A screen reader reads alternative tag names in HTML that describe graphic image content.

EXAMPLE 4 A graphic encyclopaedia's animation (dynamic object) provides a stored textual description: "A lava flow pours from the volcano, covering the town below it within seconds". If a video description of an object is available, it may not be necessary to provide an additional textual description.

7.2.6 Accessibility features should be easy to turn on and off (primary: OS)

The on/off controls for accessibility features should be easy to activate.

EXAMPLE 1 The user can activate the "Sticky keys" accessibility feature by pressing the shift key five times in succession.

EXAMPLE 2 Low-vision adjustments are shown in large type by default.

7.2.7 Safeguard against inadvertent activation or deactivation of accessibility features (secondary: OS)

Inadvertent activation or deactivation of accessibility features should be prevented.

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

7.2.8 Inform user of accessibility feature on/off status (primary: OS)

The current status of accessibility features should be available at all times.

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

EXAMPLE 2 A dialog informs the users that they have activated a "Show sounds" accessibility feature.

7.2.9 Enable persistent activation (secondary: OS/application)

When users can activate a menu, control, or other user-interface object to display additional information, software should allow that information to persist while the user engages in other tasks, until the user chooses to dismiss it ([29] in the Bibliography), if it is appropriate to the task.

NOTE Persistent display of frequently used windows and controls may be helpful for users who have physical 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.

7.2.10 Avoid seizure-inducing blink rates (core: OS/application)

The use of blinking/flashing text or graphical elements at frequencies between 10 Hz to 25 Hz should be avoided, as the probability of induced seizure is highest at these rates ([26] in the Bibliography).

7.2.11 Provide undo functionality (secondary: OS/application)

A mechanism should be provided that enables users to undo the effects of unintended actions and/or require confirmation ([29] in the Bibliography).

NOTE 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.

EXAMPLE 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.

7.2.12 Enable user control of time-sensitive presentation of information (core: application)

Whenever moving, blinking, scrolling, or auto-updating information is presented, the user should be enabled to pause or stop the presentation.

7.2.13 Clarify natural language usage (secondary: OS/application)

Additional information and/or functionality that facilitates pronunciation or interpretation of abbreviated or foreign text should be used ([37] in the Bibliography).

EXAMPLE The abbreviation “e.g.” in an article is pronounced by a text-to-speech-converter as “for example” and not as “e” “g”.

7.3 Assistive technologies

7.3.1 Use system-standard input/output (core: application)

In order to enable effective use of assistive technologies, software should use system-provided input and output methods, wherever possible. If system routines must be by-passed, system-state information should be set or obtained using system variables or system routines ([1] and [34] in the Bibliography).

EXAMPLE 1 The software moves a text 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.

EXAMPLE 3 Software reads text input using custom routines for special-purpose use, but copies that text to the system text buffer so that text is readable by assistive technology.

7.3.2 Provide object labels (core: OS/application)

Users should be enabled to access object labels stored as additional text, whether those labels are visually presented or not ([1], [34] and [37] in the Bibliography).

EXAMPLE 1 An icon depicting an eraser on a palette has no visible label. Although it is not shown, a non-visible label, such as an icon-variable name, is assigned a meaningful name (“eraser”) that may be recognized and read to the user by assistive software when the object has focus or the pointer moves over it.

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.

EXAMPLE 3 Compound objects that consist of a collection of other objects have a group label. A web-page image composed of a series of smaller image files, for example, provides a group label for a “Construction Site” instead of relying on each image label: “full view of a building”, “bulldozer”, “dump truck”, “crane”, etc.

7.3.3 Make event notification available to assistive technologies (core: OS/application)

Notification of events should be provided to assistive software ([8] and [37] in the Bibliography). Events relevant to user interactions should be available to assistive technologies. Such events include, but are not limited to, changes in object status, such as the creation of new objects, change in selection, change in position, as well as changes in attributes such as size and colour.

Users also need feedback on input events, such as key-presses and mouse button presses, and output events that the system provides, such as writing text to the screen or playing audio information (see ISO 9241-10 and ISO 9241-13).

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

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

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

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

EXAMPLE 5 When an audio is playing, assistive software that generates speech can delay providing output in order to avoid conflicting with the existing audio event.

7.3.4 Make object attributes available to assistive technologies (core: OS/application)

Information on individual object attributes should be available to assistive technologies (such as “listeners”). Such attributes may include, but are not limited to, object name, size, position and current state.

These object attributes are typically available to users by inspection/interaction. Users with certain disabilities may not be enabled to see or otherwise detect these attributes without using assistive software.

Information is made available to assistive software so that it can do what is described in the following examples.

EXAMPLE 1 Determine the word, sentence and paragraph boundaries of an area of text on the screen.

EXAMPLE 2 Determine the location of a button on the screen.

EXAMPLE 3 Determine the name of a window.

EXAMPLE 4 Obtain the foreground and background colours of a button.

EXAMPLE 5 Determine the boundaries, font and colour of an area of text on the screen.

7.3.5 Present user notification in a relevant manner (core: OS/application)

Alerts, warnings, and other user notifications of critical importance should be consistently located on the screen, and be labelled to be clearly identifiable as critical user information and be presented in a manner useful to users of assistive technologies.

EXAMPLE 1 For users who have chosen to receive auditory feedback, a beep is provided when an error message has been displayed or a system message has been updated (see ISO 9241-12:1998, subclause 5.1).

EXAMPLE 2 Critical information is presented in text, not only as an image or graph.

EXAMPLE 3 Error information presented in a window is labelled in a manner that makes clear the source and context in which the error occurred, so that a person reading from a Braille display can easily determine the message context.

EXAMPLE 4 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.

7.4 Keyboard-input configuration

7.4.1 General

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

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

7.4.2 Enable sequential entry of multiple keystrokes (core: OS)

Users should be enabled to lock or latch modifier keys (e.g. Shift, Control, Num Lock) so that multiple key combinations and key+mouse button combinations can be entered sequentially rather than by simultaneously pressing multiple keys ([29], [31] and [34] in the Bibliography).

NOTE 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 Sticky keys (available on most major platforms).

7.4.3 Provide customization of delay before key acceptance (primary: OS)

The system should enable users to individualize the delay during which a key is held down before a key-press is accepted ([29], [31] and [34] in the Bibliography).

NOTE 1 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 2 This feature cannot be active at the same time as Inter-key-press delay (see following guideline). The difference is that the acceptance delay requires a user to hold a key down for a minimum time before acceptance, but the inter-key-press delay ignores key-presses that occur before a specified delay. These are mutually exclusive because a user can press a key long enough to be accepted, but then strike another key before the inter-key-press delay time, in which case, the two requirements of inter-key-press delay and a minimum key-press hold time would be in conflict.

EXAMPLE Slow keys (available on most major platforms).

7.4.4 Provide customization of same-key double-strike acceptance (primary: OS)

Users should be enabled to individualize the delay after a keystroke, during which an additional key-press will not be accepted if it is identical to the previous keystroke. Keystrokes should not be buffered in the event that an inter-key-press acceptance delay has been set ([31] and [34] in the Bibliography).

NOTE 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.

EXAMPLE Bounce keys (available on most major platforms).

7.4.5 Provide customization of key repeat rate (primary: OS)

Users should be enabled to individualize the rate of key repeat ([34] in the Bibliography). Comment for clarity: this provides for acceptance versus the provision for delay described in 7.4.4.

7.4.6 Provide customization of post-key-press delay of repeat onset (primary: OS)

The system should enable users to individualize the minimal amount of time they can hold a key down after key-press acceptance before key repeat begins, including the option to turn off key repeat altogether ([29] and [34] in the Bibliography).

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.

EXAMPLE Repeat keys (available on most major platforms).

7.4.7 Provide keyboard control of pointer functions (core: OS)

The system should provide a keyboard alternative to standard pointing devices that enables keyboard control of pointer movement and pointer button functions ([29] and [34] in the Bibliography).

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

EXAMPLE Mouse keys (available on most major platforms).

7.4.8 Provide notification about toggle-key status (primary: OS)

The system should provide information concerning the toggle-key status to the user and/or the assistive technology ([29] in the Bibliography).

NOTE 1 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 2 Existing systems indicate a locked state with an up tone, and an unlocked state with a down tone. This includes settings made using “sticky keys” (see 3.33).

EXAMPLE Toggle keys (available on most major platforms).

7.4.9 Provide accelerator keys (secondary: OS/application)

Software should provide keyboard accelerators for frequently used features ([29] and [34] in the Bibliography).

NOTE 1 In many cases, not every feature can or needs to be mapped to an accelerator. 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, if a user were restricted to only those features.

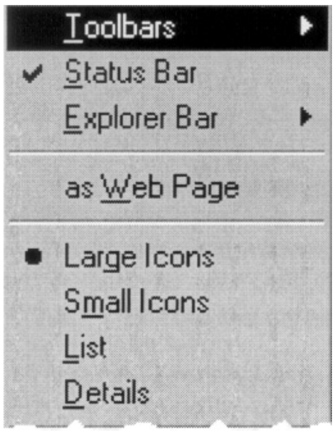
NOTE 2 Accelerators are especially important for users who interact only through a keyboard, voice-recognition systems, or who map combinations of frequently used features to macros. Users who have disabilities benefit because they can reduce time-consuming steps that would otherwise be required to activate accelerated features.

EXAMPLE User can press “Ctrl-A” to select all or “Ctrl-P” to print.

7.4.10 Provide implicit designators (secondary: OS/application)

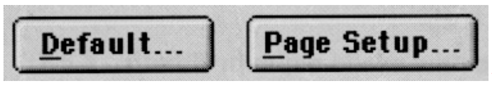
Software should provide the option to display implicit designators.

EXAMPLE 1 In the following portion of a menu:



the implicit designators are the underlined letters: “T”, “S”, “E”, “W”, “g”, “m”, “L” and “D”.

EXAMPLE 2 In the following pushbuttons:



the implicit designators are “D” and “P”.

NOTE Systems may provide the option to turn the display of implicit designators on or off.

7.4.11 Reserve accessibility key-mappings (core: OS/application)

Specific keyboard mappings reserved for accessibility should not be used for other purposes ([17], [34] and [39] in the Bibliography).

EXAMPLES

Keyboard mapping	Used for
Five consecutive clicks of shift key	On/Off for sticky keys
Shift key held down 8 s	On/Off for slow keys and repeat keys

7.4.12 Enable remapping of keyboard functions (secondary: OS/application)

Users should be enabled to re-map all keys.

NOTE Remapping is particularly valuable for accelerator and function keys ([2] and [13] in the Bibliography).

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

7.4.13 Separate keyboard navigation and activation (core: OS/application)

Keyboard navigation actions should not activate user-interface objects. An explicit activation key or key sequence should always be provided.

NOTE Keyboard navigation actions may generate events that change the user interface, such as display explanatory text on a link, or flip a page as the user scrolls through a list, but not permanently change the state of user-interface objects themselves.

EXAMPLE 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.

7.5 Software control of pointing devices

7.5.1 Types of pointing device

The term “pointing device” in this subclause refers to any physical or logical pointing device. Such devices include mice and trackballs, but they may also include head trackers, “sip & puff” systems, and many other hardware/software combinations that systems treat as a pointing device.

7.5.2 Enable the adjustment of the location of button functions (core: OS/application)

Users should be enabled to easily adjust the location of the function assigned to each pointing device button ([29] in the Bibliography).

EXAMPLE 1 A user with partial paralysis in the right arm wishes to remap the position of the mouse buttons to use it with his or her 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 his or her ability to reach them.

7.5.3 Enable multiple clicks with single key press and release (primary: OS/application)

Users should be allowed to set buttons on pointing devices to perform multiple clicks when there is a single button press and release, up to the number used for common functions.

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

EXAMPLE 1 A user with limited dexterity maps a trackball button to act as a double click.

EXAMPLE 2 A user with limited dexterity maps a trackball button to act as a triple click to select a section of text.

7.5.4 Enable button hold with a single button press and release (primary: OS/application)

Users should be allowed to adjust button-press criteria so that they are not required to hold down a pointing device button for more than the system single-click time in order to directly manipulate an object, activate a control, or maintain a view of a menu.

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

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 button.

EXAMPLE 3 Users can drag and drop objects without continuously pressing down a mouse button.

7.5.5 Enable delay of pointer-button-press acceptance (primary: OS/application)

Users should be enabled to set a minimum duration between the time a pointing-device button is pressed and when the press is accepted.

EXAMPLE A user who has tremors sets a duration sufficient to prevent tremor-induced unintentional presses from being accepted as intentional presses.

7.5.6 Enable delay of pointer movement acceptance after mouse-down event (primary: OS/application)

Users should be enabled to individualize the minimum pointer movement after a pointer-button-down event 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.

7.5.7 Enable customization of multiple-click interval and target area (primary: OS/application)

Users should be enabled to adjust the interval required between clicks in a multiple-click operation, and the distance allowed between the position of the pointer at each click, to accept the operation as a double click ([29] in the Bibliography).

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

EXAMPLE 2 A user who is moving the mouse more than the set distance and who clicks twice does not inadvertently produce a double click at a target far from the original position of the pointer.

7.5.8 Enable pointer speed and ratio adjustment (primary: OS/application)

Users should be enabled to individualize the speed or ratio at which the pointer moves in response to a movement of the pointing device ([29] in the Bibliography).

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.

7.5.9 Provide alternatives to chorded key presses (core: OS/application)

Software should provide a non-chorded alternative for any chorded key presses, whether chorded presses are on the pointing device alone or are on the pointing device in combination with a keyboard key-press ([29] in the Bibliography).

NOTE The intent here is to provide sequential alternatives to multiple simultaneous actions that 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”.

7.6 Display fonts

7.6.1 Enable font customization and legibility (core: OS/application)

Users should be enabled to set a minimum font size across all fonts, styles, backgrounds and colours presented on the display or provide the necessary information to assistive technologies to accomplish this ([1], [29] and [34] in the Bibliography).

NOTE If the operating system already provides this capability, the application may utilize them.

7.6.2 Adjust the scale and layout of objects as font-size changes (primary: OS/application)

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

NOTE This also applies to text associated with icons (see also 7.3.2.).

EXAMPLE As fonts grow, button and menu sizes grow to accommodate them. If they become large enough, the window increases in size to prevent buttons from clipping (overwriting) each other.

7.7 Displays

7.7.1 Enable users to customize viewing attributes (secondary: OS/application)

In order to increase legibility of graphics (bullets, graphics, images, etc.), software should enable users to change attributes used to determine data presentation, without changing the meaning of that data ([29] in the Bibliography).

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

EXAMPLE 1 A user who has low vision wishes to view a line graph of the stock market averages over the past 5 years. In order 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.

7.7.2 Use text characters as text, not as drawing elements (secondary: application)

In graphical interfaces, text characters should be used as text only, not to draw lines, boxes or other graphical symbols ([29] in the Bibliography).

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 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.

7.7.3 Provide access to information displayed in “virtual” screen regions (core: OS)

If the scale of the display becomes large enough to displace information from the visible portion of the screen, there should be a mechanism for accessing that information.

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

7.7.4 Enable appropriate presentation of tables (primary: application)

Tables should be created and presented in a way that enables alternative output techniques to communicate the information appropriately ([37] in the Bibliography).

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.

7.8 Colour

7.8.1 Provide alternatives to the use of colour as the sole source of information (core: OS/application)

Colour alone should not be used as the only way to convey information or indicate an action.

See ISO 9241-12:1998, subclause 7.5.

EXAMPLE 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.”

7.8.2 Provide colour palettes designed for people who have visual impairments (primary: OS/application)

Colour palettes should include default palettes designed for use by people who have visual impairments ([36] in the Bibliography).

EXAMPLE High-contrast monochrome palettes are provided. Colour-palette choices include colour combinations that provide high contrast, and other palette choices that avoid the use of colours that may confuse users who have common forms of red-green colour deficiency, cataracts, macular degeneration and other visual impairments ([34] in the Bibliography).

7.8.3 Allow users to create colour palettes (secondary: OS/application)

Users should be enabled to create their own colour palettes, including background and foreground colours.

7.8.4 Use user-determined colour settings (primary: OS/application)

Software should use colour-palette system settings individualized by users.

EXAMPLE 1 Use of colours defined by user-selected system resources.

EXAMPLE 2 If a user chooses red as the colour to represent a link, an embedded application should not override that setting to provide another colour.

7.8.5 Allow users to customize colour coding (primary: OS/application)

Except in cases where warnings or alerts have been standardized for mission-critical systems (e.g. red=network failure), the system should allow users to individualize colours used to indicate selection, process, or object state/status.

EXAMPLE A user who cannot discriminate between red and green can set the printer-status colours to be blue for OK and yellow to indicate printer problems (although note that, as per the next guideline, this colour change is expected not to be the only indication of status).

7.8.6 Provide alternatives to coding by hue (core: OS/application)

Hue should not be the only attribute used to code information ([26] and [36] in the Bibliography).

EXAMPLE 1 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.

EXAMPLE 2 If an indicator turns from green to red to show an error condition, the user can also obtain text or audio information that indicates the error condition.

7.9 Audio output

7.9.1 Use of audio output

Where software runs on systems whose hardware permits implementation of the recommendations in 7.9.2, 7.9.3, 7.9.4 and 7.9.5, the guidance in these subclauses should be followed. This will maximize software accessibility in cases where the hardware supports the recommended behaviour.

7.9.2 Enable audio customization (secondary: OS/application)

Users should be enabled to individualize attributes of audio output such as frequency, volume, speed, and sound content ([26] and [36] in the Bibliography).

NOTE The range of customization 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.

7.9.3 Default frequency range for non-speech audio (secondary: OS/application)

The fundamental frequency of task-relevant non-speech audio should occur in a range between 500 Hz and 3 000 Hz or be easily adjustable by the user into that range ([26] in the Bibliography).

NOTE Sounds in this range are most likely to be detectable by people with hearing impairments.

7.9.4 Provide specified frequency components for audio warnings and alerts (secondary: OS/application)

Alerts and other auditory warnings 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, if appropriate for the task ([26] in the Bibliography).

7.9.5 Allow users to choose visual indication of audio output (core: OS/application)

Users should be enabled to choose to have system or application task-critical audio output (including alerts) presented in visual or auditory form, or both together ([29], [34] and [36] in the Bibliography).

EXAMPLE 1 For users who have chosen to receive visual feedback, a flashing border on a dialog box is provided in place of a warning tone.

EXAMPLE 2 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.

7.10 Errors and user notification

7.10.1 Allow task-relevant warning or error information to persist (core: OS/application)

Error or warning information should persist or repeat in a suitable manner for as long as it is relevant to the performance of the task or until the user dismisses it ([29], [34] and [36] in the Bibliography).

NOTE Auditory warnings or errors need only repeat at intervals appropriate to the user's task, and do not need to persist continuously.

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