INTERNATIONAL STANDARD

ISO 9241-171

First edition
2008-07-15

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

Reference number
ISO 9241-171:2008(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

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 9241-171 was prepared by Technical Committee ISO/TC 159, *Ergonomics*, Subcommittee SC 4, *Ergonomics of human-system interaction*.

This first edition of ISO 9241-171 cancels and replaces ISO/TS 16071:2003, of which it constitutes a technical revision.

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

Guidance on software individualization and human-centred design process for interactive systems are to form the subjects of future parts 129 and 210.

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

— *Part 20: Accessibility guidelines for information/communication technology (ICT) equipment and services*

— *Part 110: Dialogue principles*

— *Part 151: Guidance on World Wide Web user interfaces*

— *Part 171: Guidance on software accessibility*

— *Part 300: Introduction to electronic visual display requirements*

— *Part 302: Terminology for electronic visual displays*

— *Part 303: Requirements for electronic visual displays*

— *Part 304: User performance test methods*

— *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 308: Surface-conduction electron-emitter displays (SED)* [Technical Report]

— *Part 309: Organic light-emitting diode (OLED) displays* [Technical Report]

— *Part 400: Principles and requirements for physical input devices*

— *Part 410: Design criteria for physical input devices*

— *Part 920: Guidance on tactile and haptic interactions*

Framework for tactile and haptic interaction is to form the subject of a future part 910.

Introduction

The purpose of this part of ISO 9241 is to provide guidance on the design of the software of interactive systems so that those systems achieve as high a level of accessibility as possible. Designing human-system interactions to increase accessibility promotes increased effectiveness, efficiency and satisfaction for people having 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 increasing 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 individualization (see ISO 9241-110), and

— offering individualized user instruction and training.

It is important to incorporate accessibility goals and features into the design as early as possible, when it is relatively inexpensive compared to the cost of modifying products to make them accessible once they have been designed. As well as providing guidance for achieving that, this part of ISO 9241 addresses the increasing need to consider social and legislative demands for ensuring accessibility by the removal of barriers that prevent people from participating in life activities such as the use of environments, services, products and information.

This part of ISO 9241 is applicable to software that forms part of interactive systems used in the home, in leisure activities, in public situations and at work. Requirements and/or recommendations are provided for system design, appearance and behaviour, as well as specific accessibility issues, thereby complementing International Standards ISO 9241-11, ISO 9241-12, ISO 9241-13, ISO 9241-14, ISO 9241-15, ISO 9241-16 and ISO 9241-17, ISO 9241-110 and ISO 14915 as well as reflecting the goals outlined in ISO Guide 71 [60]. Conforming with the aforementioned International Standard is also important if the goal of accessibility is to be achieved.

NOTE 1 While the requirements and recommendations of this part of ISO 9241 are generally applicable to all software application domains, additional detailed guidance on the accessibility of Web content (including Web applications) is available from the Web Content Accessibility Guidelines (WCAG) [53].

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 groups of people. The intended users of interactive systems are consumers or professionals — people at home, at school, 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 can 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” [50]. 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 who 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 expression *design for all* or, in North America, *universal design* [9], 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 specialized products. Accessibility as defined in this part of ISO 9241 emphasizes the goals of maximizing 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 technologies 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 can 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 pointing devices instead of mice and Braille output devices instead of a video display. There are many others. 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, platforms (including operating systems) must provide programming services to allow software to operate effectively with add-on assistive software and hardware as specified in this part of ISO 9241. 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 the 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 are in accordance with this part of ISO 9241.

NOTE 2 In this document the term “developers” is used as shorthand for *all those involved in the development of software design and creation*, which sometimes can span different collaborating or contracting organizations.

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

ISO 9241 was originally developed as a seventeen-part International 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 general 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 Technical Committee ISO/TC 159, SC 4. The revised multipart standard is structured as series of standards numbered in the “hundreds”: 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.

See Annex A for an overview of the entire ISO 9241 series.

Ergonomics of human-system interaction —

Part 171:
Guidance on software accessibility

1 Scope

This part of ISO 9241 provides ergonomics guidance and specifications 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. It addresses software considerations for accessibility that complement general design for usability as addressed by ISO 9241-110, ISO 9241-11 to ISO 9241-17, ISO 14915 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).

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

It is intended for use by those responsible for the specification, design, development, evaluation and procurement of software platforms 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 9241-14:1997, *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 9241-110:2006, *Ergonomics of human-system interaction — Part 110: Dialogue principles*

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

ISO 14915 (all parts), *Software ergonomics for multimedia user interfaces*

3 Terms and definitions

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

3.1
accelerator keys
shortcut keys
key combinations which invoke a menu option without displaying the menu on which the option appears or intermediate menus

[ISO 9241-14:1997]

3.2
accessibility
〈interactive system〉 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 recognized 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
accessibility feature
feature (etc.) that is specifically designed to increase the usability of products for those experiencing disabilities

3.4
activation
internal state with differential degrees of mental and physical functional efficiency

[ISO 10075:1991]

3.5
assistive technology
AT
hardware or software added to, or incorporated within, a system that increases accessibility for an individual

EXAMPLE Braille display, screen reader, screen magnification software, eye tracking devices.

3.6
chorded key-press
keyboard key or pointing-device 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.

3.7
closed system
system that does not allow user connection or installation of assistive technology that would have programmatic access to the full user interface

NOTE This can be because of policy, system architecture, physical constraints or for any number of other reasons.

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
contrast
〈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.)

[CIE 17.4:1987, definition 845-02-47]

3.10
cursor
visual indication of where the user interaction via keyboard (or keyboard emulator) will occur

cf. **keyboard focus cursor** (3.22), **text cursor** (3.35), **pointer** (3.30)

3.11
effectiveness
accuracy and completeness with which users achieve specified goals

[ISO 9241-11:1998, 3.2]

3.12
efficiency
resources expended in relation to the accuracy and completeness with which users achieve goals

[ISO 9241-11:1998, 3.3]

3.13
explicit designator
code or abbreviation for a menu option or control label, set apart from the name (usually to the left of it), and typed in for selection

cf. **implicit designator** (3.16)

EXAMPLE "O", "C", "S", "P", as shown in the menu in Figure 1.

O	Open
C	Close
S	Save
P	Print

Figure 1 — Examples of explicit designators

3.14
focus cursor
location cursor
indicator showing which user-interface element has keyboard focus

cf. **input focus** (3.18) and **cursor** (3.10).

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

NOTE 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).

3.15
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]

3.16
implicit designator
portion of an option name or control label used for keyboard selection

EXAMPLE "P" on a screen used for initiating a print job where the control label is displayed as "Print".

3.17
individualization
modification of interaction and presentation of information to suit individual capabilities and needs of users

3.18
input focus
in relation to a given input device, the indication of the object upon which the user directs input

[ISO 9241-16:1999]

EXAMPLE Pointer focus and keyboard focus are input foci.

3.19
keyboard emulator
software or hardware that generates input identical to that from a keyboard

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

EXAMPLE Platform-based on-screen keyboards, speech input, handwriting, wherever their output appears to applications as keystroke input.

3.20
keyboard equivalent
key or key combination that provides access to a function usually activated by a pointing device, voice input or other input or control mechanism

3.21
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.

3.22
keyboard focus cursor
visual indication of where the user interaction via keyboard (or keyboard emulator) will occur

cf. **keyboard focus** (3.21), **pointer** (3.30), **text cursor** (3.35).

3.23
label
short descriptive title for an entry or read-only field, table, control or object

cf. **name** (3.27)

EXAMPLE 1 Heading, prompt for entry field, text or graphic that accompanies and identifies a control (such as those displayed on the face of buttons), audible prompt used by an interactive voice response system.

EXAMPLE 2 “Creation Time”, as shown in Figure 2.

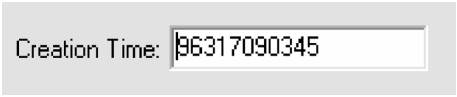

Figure 2 — Example of text field with label

EXAMPLE 3 “Pagination”, “Widow/Orphan control”, “Keep with text”, “Keep lines together”, “Page break before”, as shown in Figure 3.

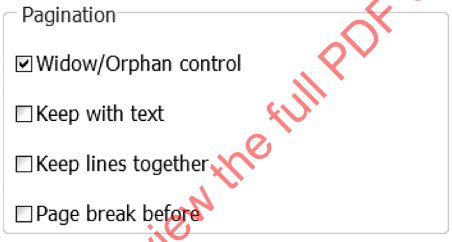

Figure 3 — Example of check box group with labels for group and each check box

EXAMPLE 4 Image of a printer in a window that the user can click to print the current document.

NOTE 1 In some applications, labels are classified as protected fields.

NOTE 2 Adapted from ISO 9241-17:1998, definition 3.4.

NOTE 3 For the purposes of this part of ISO 9241, *label* refers to the presented title for a user-interface element — in contrast with the *name* attribute, which might or might not be presented to users but is available to assistive technologies. Textual labels are often a visual display of the name.

3.24
latch
mode in which any modifier key remains logically pressed (active) in combination with a single subsequent non-modifier key-press or pointing-device button action

cf. **lock** (3.25)

3.25
lock
persistent mode in which one or more modifier keys or pointing-device buttons remain logically pressed (active) until lock mode for the key or button is turned off

cf. **latch** (3.24)

NOTE 1 Unlike *latch*, which affects only keyboard and pointing device actions, *lock* will 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 can also be turned off at other times such as at system shutdown or restart.

3.26
modifier key
keyboard key that changes the action or effect of another key or a pointing device

EXAMPLE 1 Moving the keyboard focus with the shift key held down, thereby extending the current selection in the direction of cursor movement rather than merely moving the position of the cursor.

EXAMPLE 2 Pressing “C”, to obtain the input of that character, pressing “Ctrl+C” to obtain the “Copy” function.

3.27
name
word or phrase associated with a user-interface element and that is used to identify the element to the user

cf. **label** (3.23).

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

NOTE 2 The name attribute might or might not be presented to users but is available to assistive technologies — in contrast with *label*. For the purposes of this part of ISO 9241, *label* refers to the presented title for a user-interface element. Textual labels are often a visual display of the name.

NOTE 3 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 4 Names are not to be confused with internal identifiers (ID), which can be used by software and might not be designed to be understood by a human.

3.28
natural language
language that is, or that was, in active use in a community of people, the rules of which are mainly deduced from the usage

3.29
platform software
software that interacts with hardware or provides services for other software

EXAMPLE Operating system, device driver, windowing system, software toolkit.

NOTE 1 A browser can function both as an application and as platform software.

NOTE 2 For the purposes of this part of ISO 9241, *software* refers to both platform software and application software.

3.30
pointer
graphical symbol that is moved on the screen according to operations with a pointing device

NOTE Users can interact with elements displayed on the screen by moving the pointer to that location and starting a direct manipulation.

[ISO 9241-16:1999, 3.15]

3.31
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.

3.32
pointing device
device that translates a human controlling operation to a controlling operation on the display

NOTE 1 Depending on the applied technology, not only machine devices but also parts of the human body (e.g. fingers, arms) can currently be used as pointing devices.

[ISO 9241-16:1999]

NOTE 2 Pointing devices typically have buttons that are used to activate or manipulate user-interface elements.

NOTE 3 Almost any hardware can be used to control a **pointer** (3.30) with the appropriate software.

3.33
satisfaction
freedom from discomfort, and positive attitudes towards the use of the product

[ISO 9241-11:1998, 3.4]

3.34
screen reader
assistive technology that allows users to operate software without the need to view the visual display

NOTE 1 Output of screen readers is typically text-to-speech or dynamic Braille output on a refreshable Braille display.

NOTE 2 Screen readers rely on the availability of information from the operating system and applications, such as the name or label of the user-interface element.

3.35
text cursor
visual indication of the current insertion point for text entry

cf. **pointer** (3.30), **focus cursor** (3.14).

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, 3.1]

3.37
user interface
UI
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]

3.38
user-interface element
user-interface object
entity of the user interface that is presented to the user by the software

EXAMPLE Text, graphic, control.

NOTE 1 User-interface elements can be interactive or not.

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

NOTE 3 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.

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. The more accessible a design, the wider the range of people who will find it usable.

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 effectiveness, efficiency and satisfaction of users working with a product or interacting with an environment for the widest range of users. 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 can form 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 Principles for designing accessible software

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 particular context of use — to increase the number of people who are able to use the software. The guidance provided in this part of ISO 9241 is applicable at any stage in the development of an interactive system.

Designing accessible software should adhere to the following principles.

— **Equitable use**

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

— **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.

— **Robustness** (WCAG 2.0 Principle No. 4) [53]

Software should be designed to be as *robust* as possible to allow it to work with current and future assistive technologies. 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.

EXAMPLE Systems that provide built-in screen magnification can enable many more users to read the text and see the images that are presented. However, if the necessary integration information is available, users can attach the screen magnification program of their choice to suit their particular needs.

The development of solutions that will result in accessible software should be based upon the application of ergonomic design guidance: ISO 9241-12, ISO 9241-13, ISO 9241-14, ISO 9241-15, ISO 9241-16, ISO 9241-17 and ISO 9241-110, and ISO 14915-1, ISO 14915-2 and ISO 14915-3, shall be consulted. The guidance in these International Standards is also included in the provisions of other standards and publications such as the Web Content Accessibility Guidelines 2.0 (WCAG) [53], universal design principles/design for all (DFA) [9], which specifically aim to increase accessibility. Issues that represent ergonomics principles and good 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, ISO 9241-110, WCAG 2.0 Principle No 3);

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

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

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

The additional guidance provided in this part of ISO 9241 specifically addresses issues that arise when providing design solutions that will satisfy the needs of users with a great variety of capabilities.

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 part of ISO 9241 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 can be

experienced intermittently and appear unpredictably or very suddenly. In addition, a disability can be experienced as an entirely unique event for an individual (e.g. having a broken arm). Annex D 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 can arise. In addition, limitations on activities can 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 can range from declining ability to resolve small detail to having been blind from birth. Different provisions in the design of the interactive system could 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 recognize that people can experience multiple disabilities. Guidance that is appropriate to addressing a specific type of disability might 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 people who are 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 individualized to the user and their task.

7 How to use this part of ISO 9241

7.1 General

In order to achieve accessibility, it is necessary to provide support in different parts of the system of software, which includes platform software (the operating system and associated layers, and toolkits) and other software (such as most applications) that 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. Similarly, accessibility characteristics implemented by the platform can require cooperation from layers running on top of them in order to be fully effective. The majority of the requirements and recommendations given in Clauses 8, 9, 10 and 11 demand that the issue be addressed at more than one level of the software system if the particular requirement or recommendation is to be satisfied.

These dependencies can occur in relation to different layers in the platform (e.g. window management on top of process management and screen drawing, which are on top of hardware drivers) and in relation to the applications that are mounted on the platform. Applications themselves can have layers that result in dependencies arising within different levels of the application.

Most of the provisions given in this part of ISO 9241 apply to all software that implements or contributes to the software user interface — regardless of whether or not it is part of the platform. Some are only applicable to portions of platform software (such as guidelines about low-level input, window management or system-wide behaviour): for example, the platform is the general means by which accessibility features that involve control of hardware devices, in particular those involving input, are implemented and controlled. Similarly, other provisions might only apply to software that displays user-interface elements, generates sounds, or exhibits other specific behaviours. In these cases, the applicable layers or type of software is indicated in the text of the requirement or recommendation, and in some cases are further elaborated upon in accompanying Notes.

7.2 Conformance

Conformance with this part of ISO 9241 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 Annex B.

Users of this part of ISO 9241 shall evaluate the applicability of each requirement (a “shall” statement) and should evaluate the applicability of each 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 (“may” statements give permission). If a product is claimed to have met the applicable recommendations in ISO 9241-171, the procedure used in establishing requirements for, developing and/or evaluating, the software accessibility shall be specified. The level of specification of the procedure is a matter of negotiation between the involved parties.

ISO 13407:1999, 7.2, and ISO 9241-11:1998, 5.3, shall be consulted for guidance on the identification and specification of the context of use; see Annex C for guidance on assessing applicability.

Annex C provides a means both for determining and recording the applicability of all the requirements and recommendations and 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 shall be in accordance with all requirements of Clauses 8, 9, 10 and 11, except for those given in 8.6.

8 General guidelines and requirements

8.1 Names and labels for user-interface elements

8.1.1 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.

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

NOTE 2 If some names are missing, assistive technology might be unable to sufficiently identify or operate the user-interface elements for the user.

NOTE 3 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).

EXAMPLE 3 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 4 When a script or object hosted within a Web browser attempts to set the window’s title to a string that is already used by another of the browser’s windows, the browser modifies this string to be unique.

NOTE 4 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.

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

NOTE 5 Such elements can be containers that serve to group one or more sub-elements. In a typical graphical user interface, examples of UI (user interface) 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 6 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 (“Construction site”) in addition to the names of the individual component images (“building”, “bulldozer”, “dump truck”, “crane”, etc.).

8.1.2 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.

EXAMPLE 1 The name of a checkbox is “Gender” and not “Gender checkbox.”

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

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

EXAMPLE 2 A control is listed in the product documentation as “the Print button”; therefore 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 3 Dialog boxes or windows have meaningful names, so that a user who is using speech output because he/she cannot see the screen gets appropriate contextual information.

8.1.3 Provide unique names within context

Each name of a user-interface element specified by software developers 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 document, location or person) can be provided with the name of that entity even if the name is not unique in its context.

EXAMPLE 1 A form displays areas containing fields with a customer’s home and business details. Each area has an associated “Change” button. Rather than duplicate the name, the buttons are named “Change Home” and “Change Business”.

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 A user opens a second window of the same document using a word processing application. Both windows are of the same document and both are editable. The word processor adds a ":1" to the end of the document name to form the name of the first window. It names the second window with the same document name, except that it appends a ":2" to the end of the second window name so that the two windows have unique names.

EXAMPLE 4 When a script or object hosted within a Web browser attempts to set the window's title to a string that is already used by another of the browser's windows, the browser modifies this string to be unique.

8.1.4 Make names available to assistive technology (AT)

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.

NOTE 1 In a platform that does not provide a standard service for the association of names and elements, application developers document how assistive technologies can access that information.

NOTE 2 Services for the association of names and elements are part of those described in 8.1.6.

8.1.5 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 users (either by default or at the user's request).

NOTE Standard components of the user interface are components that are provided by the platform and that look and behave the same across applications.

EXAMPLE 1 In an application, the window scroll up and scroll down buttons do not have a label or pop-up text because these buttons are standard across applications on the platform. At the bottom of the scrollbar, however, are special "jump to next find" and "jump to last find" arrows that do have pop-up text that describes their function.

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

8.1.6 Provide names and labels that are short

Each name or label of a user-interface element specified by the software developers should be short enough to be rendered succinctly.

NOTE 1 If developers are encouraged to put the most distinctive parts of the name first, 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 document, location or person) can be provided with the name of that entity even if the name is too lengthy or cryptic to be read easily.

NOTE 3 Using brief labels also benefits users of auditory, visual and tactile output.

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 software developers.

8.1.7 Provide text label display option for icons

Software should allow users to choose to display either the icon images, the icon images with text labels or the icon text labels only.

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

8.1.8 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 (see also ISO 9241-12:1998, 5.9.4 and 5.9.6).

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

NOTE This helps AT correctly associate the labels with their corresponding controls, and helps users of screen magnification software know where to look for a label or control.

8.2 User preference settings

8.2.1 Enable individualization of user-preference settings

When the software enables the user to set personal preferences, these settings should be easily adjustable.

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

NOTE 1 System-wide user preference settings provided by the platform need to be used in addition to any preference settings for product-specific options.

EXAMPLE 2 A software application allows a user with cognitive disabilities to choose the number and size of icons displayed at any one time.

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

EXAMPLE 3 A user chooses preference settings through a graphical user interface, rather than directly editing the configuration files.

NOTE 3 Business considerations related to consistency of operations, performance-based considerations, safety, privacy, and security concerns can all lead to some necessary restriction by system administrators of the user's capability to modify the behaviour and appearance of user-interface elements in certain contexts. Administrators need to show restraint in limiting user control. Not all options/preferences settings are appropriate for such administrative control.

NOTE 4 Administrators within this business environment can make specific permission profiles for users that require more flexibility in their options/preference settings for usability and accessibility.

8.2.2 Enable adjustment of attributes of common user-interface elements

Software should enable users to adjust the attributes of common user-interface elements if applicable to the task.

NOTE 1 Common attributes for a visual interface could include, but are not limited to, font type, font size, and font colour. For an auditory interface they could include, but are not limited to, aural cue type, rate, volume, pitch, position in 3D audio space. For a tactile interface they could include, but are not limited to, haptic object size, texture, xy- or xyz-position, pressure sensitivity, solidity.

NOTE 2 Platform software often supports these options for standard user-interface elements it provides. To enhance user experience, applications can use the settings defined at the platform level.

EXAMPLE Software retains user preference for window size and location between sessions.

8.2.3 Enable individualization of the user interface look and feel

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

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

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

8.2.4 Enable individualization of the cursor and pointer

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

NOTE 1 Platform software often supports these options for standard cursors and pointers it provides, and software using these cursors and pointers can automatically comply with this requirement.

NOTE 2 The ability to set the cursor to non-blinking is important for users with attention deficits, who may be easily distracted.

NOTE 3 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.

EXAMPLE 1 Users with low vision can change a text 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 keyboard 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.

8.2.5 Provide user-preference profiles

Software should enable users to create, save, edit and recall profiles of preference settings, including input and output characteristics, without having to 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 can be advisable.

NOTE 2 It is often useful to be able to access the preference settings over a network. Doing this in a secure way would preserve privacy especially for people who are worried about revealing the fact that they have a disability.

NOTE 3 It is preferable 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 For a person completing an on-line process who has to make adjustments to the accessibility feature to reduce errors, restarting the operating system or the user agent would cause loss of work.

8.2.6 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 could find a system difficult or impossible to use without 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 the 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 can provide a general mechanism for transferring their preference settings; in such cases, software might 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 USB (universal serial bus) drive onto a new computer.

EXAMPLE 3 A user's preference settings are loaded from a smart card onto a new system.

8.2.7 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;

— 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;

— 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 log-on prompt requires the user to enter their password within 30 s. The remaining time is shown on the screen and a control provided to stop the time decrementing.

8.3 Special considerations for accessibility adjustments

8.3.1 Make controls for accessibility features discoverable and operable

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

NOTE Features are considered *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 (see Annex E) 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 individualizing low-vision options are shown in large type by default.

8.3.2 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.

8.3.3 Avoid interference with accessibility features

Software shall not disable or interfere with the accessibility features of the platform.

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

8.3.4 Inform user of accessibility feature "On"/"Off" status

Software should allow the user to identify the current status of accessibility features.

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.

8.3.5 Inform user of accessibility feature activation

When an accessibility feature can be activated unintentionally, software should inform users and provide an opportunity to accept or cancel the activation.

NOTE 1 The alert for an individual accessibility feature could be disabled at user request but would be "on" by default.

NOTE 2 It is good practice to provide an alert whenever the SlowKeys [1)] feature (see Annex E) is activated via a keyboard shortcut method, since an accidental activation could lead an uninformed user to believe that the keyboard was broken. Regular users of SlowKeys, however, prefer to have a way to defeat the alert so that they do not have to dismiss the alert each time they turn their SlowKeys feature on.

8.3.6 Enable persistent display

When users can activate a menu, control, or other user-interface element to display additional information or controls, software should allow that information or control to persist while the user engages in other tasks, until the user chooses to dismiss it [44], 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. By doing so, this menu command is persistently displayed.

8.4 General control and operation guidelines

8.4.1 Enable switching of input/output alternatives

Platform software should enable users to switch among the available 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.

---

1) SlowKeys™ is a trademark of the University of Wisconsin. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of the product named. Equivalent products may be used if they can be shown to lead to the same results.

EXAMPLE 1 A person who is blind 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 One user points with a mouse to an icon for a document on their screen and says “print” to print the document while another clicks on the document and keys in CTRL+P to print it. Still another uses the “Print” item in the “File” menu to print the document.

EXAMPLE 3 To implement a setting change, an application must restart, but it restores all data including position of the keyboard focus cursor.

8.4.2 Optimize the number of steps required for any task

Software should be designed to optimize 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 determine the optimal design.

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

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

8.4.3 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 [44].

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 that undo operations themselves can be undone.

NOTE 5 However, this might not be possible for such interactions as operations causing fundamental transformation of logical or physical devices, or could involve a data exchange with third parties that are out of the software’s control, etc.

NOTE 6 The default configuration can provide a confirmation step for any action that the user cannot undo with a single “Undo” command.

NOTE 7 Software could allow the user to disable the confirmation for specific actions.

EXAMPLE 1 A user with Parkinson’s disease might inadvertently input a sequence of keystrokes that activates several dialogues that then need to be undone. The use of several steps of the undo function permits 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.

8.4.4 Provide alternatives when assistive technology is not operable

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

NOTE System start-up and restart include operations prior to the stage where the user's accessibility aids and preference settings are available. They can be either accessible or non-interactive.

EXAMPLE 1 A computer is configured with a small "boot loader" that lets the user 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 AT, 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.

8.4.5 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-ROM and DVD.

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 the on-screen keyboard to instruct the operating system to eject the disc, so that it will not interfere with the computer's next restart.

8.4.6 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.

NOTE 3 "Cut" operations can also be provided as they enable faster user work.

8.4.7 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 onto the clipboard can help users with disabilities avoid awkward, slow, or error-prone manual input of that text into other locations.

EXAMPLE 1 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.

EXAMPLE 2 An operating system provides a function whereby the user can hold down the "Control" and "Alt" keys and select any text that was drawn to the screen using the OS (operating system) text drawing routines.

8.4.8 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.

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 allows cycling 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.

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

8.4.9 Allow warning or error information to persist

Software shall ensure that error or warning information persists or repeats in a suitable manner as long as the source of the error or warning remains active, or until the user dismisses it [44], [49], [51].

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

8.4.10 Present user notification using consistent techniques

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

EXAMPLE 1 Following a “beep”, a positioning message is provided in a consistent place that allows the user having 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.

NOTE A screen reader can be programmed to read a message automatically as long as it appears in a consistent manner in a particular screen location.

EXAMPLE 3 Notification that a form field is mandatory or read-only is displayed on the status bar. On tabbing to the control, the message is automatically picked up by a screen reader that has been programmed to monitor this area of the window.

8.4.11 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 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 might or might not be comprehensible to the user.

8.4.12 Facilitate navigation to the location of errors

When software detects that users have entered invalid data, it should notify them in a way that allows the users to identify and navigate easily to the location of the error.

NOTE If the keyboard input focus is moved unexpectedly when software detects an error and not put back afterward, a screen reader user will be disorientated, and could find it difficult and time-consuming to find the location of the error in order to correct it.

EXAMPLE The user is notified of an error using an informational dialog box. When the dialog is dismissed, the keyboard input focus is placed at the location of the error, ready for the user to correct it.

8.5 Compatibility with assistive technology

8.5.1 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 only apply to systems that allow installation of assistive technology or where AT will be installed in conjunction with the software. They are not applicable to closed systems (see 7.2).

8.5.2 Enable communication between software and AT

Platform software shall provide a set of services that enable assistive technologies to interact with other software sufficient to enable the requirements or recommendations of 8.5.5, 8.5.6, 8.5.7, 8.5.8, 8.5.9 and 8.5.10 to be complied with or followed. (See references [26] and [41]).

If accessibility services are provided by the platform on which they are run, 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. An application can use these services to provide information about its user-interface elements and automation facilities to other software.

NOTE 2 AT could be running on the same system as the software or on a separate system.

EXAMPLE 1 A screen reader uses an accessibility service to query information about a non-standard user-interface element that appears on screen.

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

8.5.3 Use standard accessibility services

Software that provides user-interface elements shall use the accessibility services provided by the platform to cooperate with assistive technologies. If accordance with 8.5.5, 8.5.6, 8.5.7, 8.5.8, 8.5.9 and 8.5.10 is not possible by these means, the software shall use other services that are supported and publicly documented, and that are 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 software only has to take care of using the accessibility services when it uses non-standard user-interface elements.

NOTE 2 AT could be running on the same system as the software or on a separate system.

EXAMPLE 1 An application having non-standard user-interface elements uses the accessibility services of the operating system to provide information about the name, presentation 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. It can inform about the keyboard focus cursor position, the character, word or sentence at the text 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 AT programs for that platform, and in cooperation with them design, implement and publish a communication mechanism that each product implements.

8.5.4 Make user-interface element information available to assistive technologies

Software shall provide assistive technology with information about individual user-interface elements, as specified in 8.5.3, except for 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, keyboard 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 might 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 8.1 for more information on the name property and the relationship between an element and its visual label.

NOTE 5 See 8.5.7 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 him or her, 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 his/her screen reader to tell where he/she is working. The screen reader uses accessibility services to ask the current application for the identity 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 him/her 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 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 his/her screen reader to voice the word and character at the current insertion point, and the text that is currently selected.

EXAMPLE 5 When displaying tabular data or data in columns, the application provides AT with information about the data, including any row or column names.

EXAMPLE 6 A user executes a keyboard command asking his/her 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 7 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 is able to determine that the control's name is "Save" even though it visually appears as a picture of a floppy disk.).

EXAMPLE 8 Developers build an application using standard controls provided by the operating system. Because those controls already include support for the platform's accessibility services, developers comply with this guideline by providing names and other attributes for those controls. They make sure their pre-release testing includes users who rely on AT, who verify that the application works with their products.

EXAMPLE 9 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 10 Software provides AT with information about a scroll bar, including its type ("Scroll Bar"), name ("Vertical"), value ("47%"), size and location. The application also provides information about the individual components of the scroll bar that can be independently manipulated, including the "Up", "Down", "Page Up" and "Page Down" buttons, and position indicator. This allows users to click, drag and otherwise manipulate those components using speech recognition programs.

EXAMPLE 11 Software does not bother to provide assistive technology with information about individual lines that are used to visually represent a control. AT can determine the control's boundaries by querying its size and location attributes, so that it does not need to rely on the position of individual drawing elements.

EXAMPLE 12 A user with a haptic or tactile disability can understand the tactilely presented message because the message is also available in electronic text.

8.5.5 Allow assistive technology to change keyboard focus and selection

Software shall allow assistive technology to modify keyboard focus and selection attributes of user-interface elements, as specified in 8.5.3.

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 keyboard focus and selection to that user-interface element. The application and operating system allow the speech recognition software to do this directly, because it might not be clear to the speech recognition software how to move the keyboard focus to the user-interface element using simulated keystrokes or mouse movements.

8.5.6 Provide user-interface element descriptions

Where tasks require access to the visual or audible content of user-interface elements beyond that which the role and name attributes provide, software shall provide descriptions of those objects. These descriptions shall be meaningful to the user and available to assistive technology through a standard programmatic interface as specified in 8.5.3, whether those descriptions are presented or not [44].

NOTE 1 In contrast with the label attribute that names a user-interface element (see 8.1), and the role attribute that identifies its function, the description needs to 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 can in fact have an information function, such as acting as a separator, an icon, a visual label, etc. In such cases, the element needs to be provided with role and/or label attributes as specified in 8.5.4.

NOTE 3 Users who have low vision or are blind can use software presenting 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 AT.

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 labelled "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 the 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 encyclopaedia'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".

8.5.7 Make event notification available to assistive technologies

Software shall provide assistive technology with notification of events relevant to user interactions, as specified in 8.5.3.

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 keyboard 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 pushbutton to gain keyboard 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 AT 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 is notified and informs the user who cannot see the status light on the keyboard.

8.5.8 Allow assistive technology to access resources

If mechanisms exist, software should provide assistive technology with access to shared system resources on the system where the technology is installed or directly connected.

NOTE Such resources include, but are not limited to, processor time, space on the display, control of and input from the pointing device and keyboard, and system-wide accelerator keys. This is important so that the user is not prevented from effectively using AT 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 they both are given access to the shared keyboard resources (and the user has configured 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 that 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.

8.5.9 Use system-standard input/output

Software shall use standard input and output methods provided by the platform, or, if this is not possible, make equivalent information available, as specified in 8.5.3.

NOTE These capabilities are also useful in enabling automated testing applications, pervasive macro/scripting facilities and other software that works on behalf of the user.

EXAMPLE 1 Software moves a keyboard focus 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.

8.5.10 Enable appropriate presentation of tables

When presenting information in the form of tables or multiple rows or columns, information about layout, row and column headings, and explicit (presented) relationships among the data presented, shall also be communicated to assistive technology, as specified in 8.5.3. [52].

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

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

Platform 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 could move the mouse pointer to the general vicinity of a target, and then fine-tune the position using the alternative pointing device. There can 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.

8.5.12 Allow assistive technology to monitor output operations

Platform software shall provide a mechanism that allows assistive technology to receive notification about standard output operations and to identify the source and the original data associated with 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 be 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, 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 be notified about those drawing operations, so that it can keep track of the shapes, text and pictures visible in the image.

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

8.5.13 Support combinations of assistive technologies

Software should enable multiple assistive technologies to operate at the same time.

NOTE 1 Compatibility between different assistive technologies is the responsibility of the AT themselves.

NOTE 2 This provision includes cases where multiple software assistive technologies are connected in series or in parallel and cases where software can control the operations of various devices including hardware assistive technology.

EXAMPLE An operating system allows users to install multiple assistive technologies that can inject or filter keyboard input.

8.6 Closed systems

8.6.1 Read content on closed systems

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

8.6.2 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 keyboard focus, status or content audibly announced.

8.6.3 Operable through tactilely discernable controls

Software that is on, or intended for, installation on closed systems shall provide at least one mode where all functionality can be achieved through devices that do not require vision.

EXAMPLE Touchscreen software is designed so that all functionality can also be achieved through a keypad with keys that are easy to feel.

NOTE If software is designed to operate via keyboard it would satisfy this requirement, unless it is known in advance that the keyboard requires vision to operate, such as an on-screen keyboard, or software is specifically designed for a device with a flat membrane keyboard.

8.6.4 Pass through of system functions

Software that is on, or intended for, installation on closed systems shall pass through or implement the platform's accessibility features.

NOTE 1 "Implement the … accessibility features" means provision of similar accommodation to the accessibility features as part of the software.

NOTE 2 Accessibility features in the platform software that do not apply to the platform hardware (e.g. a keyboard feature in a kiosk that does not use a keyboard) are not considered accessibility features of the platform, and therefore software is not expected to pass through or implement those 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 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 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 pointing-device button functions, in parallel with the standard pointing device [44], [49].

NOTE 1 This feature is commonly called *MouseKeys* [2] (see Annex E).

NOTE 2 This feature 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 work 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 includes pointing device control of latching and locking of key presses.

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.

9.1.5 Provide system-wide spell-checking tools

Platform software should provide system-wide support for spelling assistance by indicating probable errors and providing suggestions when they are known. Except where the task involves the testing of the user's ability to spell, application software should support the system spell-checking service; or, where it is not provided by the platform software, the application software should provide this functionality for its own content.

NOTE 1 The ability to automatically check spelling is not possible for every language.

---

2) MouseKeys™ is a trademark of the University of Wisconsin. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of the product named. Equivalent products may be used if they can be shown to lead to the same results.

NOTE 2 Spelling is a problem for many users including people with text disabilities such as dyslexia.

EXAMPLE 1 A user's input in a textbox is checked for spelling using the operating system's spell-checking service.

EXAMPLE 2 The user of a text editor that does not provide a spell-checking feature uses the operating system's spell-checking service.

9.2 Keyboard focus

9.2.1 Provide keyboard focus and text cursors

Software shall provide a keyboard focus cursor that visually indicates which user-interface element has the keyboard focus at a given moment, as well as a text cursor to indicate the focus location within a text element.

NOTE The availability of this information to assistive technology is covered under 8.5.

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 text cursor (a flashing 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.

9.2.2 Provide high visibility keyboard focus and text cursors

Software shall provide at least one mode where keyboard focus cursors and text cursors shall be visually locatable by people with unimpaired vision at a distance of 2,5 m when software is displayed on a 38 cm (15 inch) diagonal screen at 1024 × 768 pixel resolution, without moving the cursor.

EXAMPLE 1 The software provides the option of having a thick rectangle of contrasting colour that moves to, and outlines, the control or field that has keyboard focus.

EXAMPLE 2 The software provides an option of having bright yellow triangles extend from the top and bottom of the text cursor.

9.2.3 Restore state when regaining keyboard focus

When a window regains focus, software should restore the keyboard focus, selection and active modes to the values they had before the window lost the focus, except when the user explicitly requests otherwise.

NOTE 1 This is important because, if keyboard focus is not retained when a window regains focus via keyboard navigation, a keyboard user must press many keystrokes to return to the previous location and selection.

NOTE 2 Some user actions can return keyboard focus to a window and then move the keyboard focus to a specific element within that window automatically or change the state of the document.

EXAMPLE 1 Keyboard focus is currently on the third button in a window until focus is switched to another window. When the focus returns to the original window, keyboard focus is returned to the third button in that window.

EXAMPLE 2 During editing of the contents of a spreadsheet cell, keyboard 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 text cursor is at the same end of the selected text as it was before the window lost focus.

EXAMPLE 3 On some platforms, the user can position the pointer over a window that does not have the keyboard focus, and click on a control in that window, thereby moving the keyboard focus to that window and then to the control that the user clicked.

9.3 Keyboard input

9.3.1 General

Although the provisions of this subclause refer to *keyboard input*, the source of such input can be a variety of software and hardware alternative input devices.

In this section, “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 of carrying out all tasks using only a non-time dependent keyboard (or keyboard-equivalent) input.

NOTE 1 Meeting this requirement has a particular benefit to a large number of people with different disabilities and also enhances usability for people without disabilities.

NOTE 2 This includes, but is not limited to, editing text and other document components, as well as navigation to, and full operation of, all controls, and not having the keyboard focus become trapped on any interface element.

NOTE 3 Platform-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 watercolour painting application where the darkness is dependent on the time the pointer spends at any location is exempt because the task requires time-dependent analogue input.

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

NOTE 4 Use of the MouseKeys feature does not satisfy this requirement because it is not keyboard-equivalent to the application; it is 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 user-interface elements. If there are multiple ways to perform a task, only one of them needs to be keyboard operable; although it is best if all possible forms are keyboard-operable.

NOTE 6 This does not preclude and ought 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 can 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 (see 9.3.17).

EXAMPLE 4 A user uses the tabulation key to navigate to and from a list, and uses the 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 design (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 keyboard focus.

EXAMPLE 7 An application in a PDA (personal digital assistant) 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 types in 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, "Ctrl", "Command", "Alt"/"Option", depending on the operating system) so that multiple key combinations and key-plus-mouse button combinations can be entered sequentially rather than by simultaneously pressing multiple keys [44], [46], [49].

NOTE 1 This feature is commonly called *StickyKeys* [3] (see Annex E).

NOTE 2 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 3 This allows users who have physical impairments a means to enter combination key commands (e.g. "Ctrl+C", "Ctrl+Alt+Delete") by pressing one key at a time.

EXAMPLE A graphics program allows the user to modify mouse clicks by holding down the "Delete" key. Because "Delete" 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 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 2 s [44], [46], [49].

NOTE 1 This feature is commonly called *SlowKeys* [3] (see Annex E).

NOTE 2 In most cases, this feature is supported automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

NOTE 3 A common range for a key acceptance delay feature is 0,5 s to 5 s.

NOTE 4 This feature allows users who have limited coordination, and who could 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 5 The BounceKeys [3] feature (9.3.5) will have no effect if SlowKeys is active.

9.3.5 Provide adjustment of same-key double-strike acceptance

Software 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 0,5 s [46], [49].

NOTE 1 This feature is commonly called *BounceKeys* (see Annex E).

NOTE 2 In most cases, this feature is supported automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

---

3) SlowKeys™, StickyKeys™, and BounceKeys™ are trade marks of the University of Wisconsin. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of the product named. Equivalent products may be used if they can be shown to lead to the same results.

NOTE 3 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 4 BounceKeys will have no effect if the SlowKeys feature (9.3.4) is active.

NOTE 5 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

Software should enable users to adjust the rate of key repeat down to one per 2 s.

NOTE 1 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.

NOTE 2 In most cases, this feature is supported automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

9.3.7 Provide adjustment of key-repeat onset

Software 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 2 s.

NOTE 1 This prevents users whose reaction time is slow from producing unwanted repeated characters by holding down a key long enough to unintentionally initiate key repeat.

NOTE 2 In most cases, this feature is supported automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

9.3.8 Allow users to turn key repeat off

Software that provides key repeat shall enable users to turn off the key repeat feature.

NOTE 1 This feature prevents users with very slow reaction time from producing unwanted repeated characters while holding down a key.

NOTE 2 In most cases, this feature is supported automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

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 [44].

NOTE 1 This feature is commonly called *ToggleKeys* [4] (see Annex E).

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 Subclause 8.6.7 specifies that software is required to notify assistive technology of these status changes.

---

4) ToggleKeys™ is a trademark of the University of Wisconsin. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of the product named. Equivalent products may be used if they can be shown to lead to the same results.

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 (function) and F4 keys to cycle between three different projection states [LCD (liquid crystal display), CRT (cathode ray tube) 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 [44], [49].

NOTE 1 In many cases, not every feature can or need be mapped to an accelerator key. The choice of which features to map to accelerator keys can be made by determining those features that 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 keys that are available to be used as accelerators somewhat depend on the conventions of the platform and the language of the user interface.

EXAMPLE The user can press "Ctrl+C" to copy, "Ctrl+V" to paste or "Ctrl+P" to print.

9.3.11 Provide implicit or explicit designators

Software should provide implicit or explicit designators that are displayed by default for all user-interface elements that take input and have visible textual labels, within the limits of characters that can be typed and the conventions of the platform. The implicit and explicit designators should be unique within their context or the user should be able to choose between them without carrying out any unintended action.

NOTE 1 This does not preclude providing an option to turn off the implicit and/or explicit designators.

NOTE 2 Implicit and explicit designators are restricted to the set of characters that can be displayed and typed, and therefore it might not be possible to provide designators when there are a large number of labelled elements. In such cases, designators need to be provided for the most commonly-used elements.

NOTE 3 Including a very large number of controls in a single form often results in a loss of usability, in addition to making it impossible to provide unique designators.

EXAMPLE 1 In the portion of menu shown in Figure 4, the implicit designators are "T", "S", "E", "W", "g", "m", "L" and "D".

Figure 4 — Example of implicit designators in a menu

EXAMPLE 2 In the pushbuttons shown in Figure 5, the implicit designators are "D" and "P".

Figure 5 — Examples of implicit designators in pushbuttons

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

NOTE 4 For further guidance, see ISO 9241-14:1997, Clause 7.

9.3.12 Reserve accessibility accelerator key assignments

The accelerator 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 accelerator key assignments

Accelerator key	Used for [a]
Five consecutive clicks of shift key	"On"/"Off" for StickyKeys
Right shift key held down 8 s	"On"/"Off" for SlowKeys and RepeatKeys

[a] See also Annex E.

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

9.3.13 Enable remapping of keyboard functions

Platform software should allow users to reassign the mapping of all keys unless restricted by hardware. Software running on such systems should support these remapping.

NOTE 1 Remapping is not the same thing as reassigning a function to a different key in an application (see 9.3.12). Remapping globally changes which logical key is associated with each physical key on the keyboard.

NOTE 2 In most cases, this is met automatically when software uses the standard keyboard input services provided by the platform software (and does not override this feature of the services).

EXAMPLE 1 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.

EXAMPLE 2 In order to correctly support keyboard remapping at the operating system level, an application uses the platform functions to read “virtual keys” rather than “scan codes” that are associated with physical keys.

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 would have to 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.

NOTE 3 Software does not meet this provision if a user cannot exit a data entry field without entering or change data because moving the keyboard focus to the field causes an effect (triggering a mandatory data entry mode). See also 8.1.

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

EXAMPLE 2 In addition to using the mouse to make selections from a list, the user can also use the arrow keys to move through items in a list, selecting them by hitting the space bar. In lists that allow multiple selections, the user can hold down the control key in combination with the arrow keys to move through the list, hitting the space bar for each item that the user wants to select.

9.3.15 Follow platform 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 and 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 platform conventions.

NOTE 4 Keyboard conventions can 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 when 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 this follows the convention established by the operating system.

9.3.16 Facilitate list and menu navigation

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

NOTE 1 Wrapping with auditory and visual indication of rollover is one strategy; using the “Home” and “End” keys are another. Often, both are provided.

NOTE 2 When navigational order is circular, an alert signal is 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 “Page Up”/PgUp” and “Page Down”/“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 Facilitate navigation of controls by grouping

Where there are large numbers of navigable controls, the controls should be grouped for ease of navigation.

9.3.18 Arrange controls in task-appropriate navigation order

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 can be the only order in which they can use controls.

EXAMPLE As a user presses the tabulation key, the keyboard focus cursor moves to a task-appropriate group of radio (or option) 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 an arrow key.

9.3.19 Allow users to customize accelerator keys

Software should enable users to customize the assignment of accelerator keys to actions.

EXAMPLE 1 An application allows the user to create macros that carry out one or more actions and to assign these macros to accelerator keys.

EXAMPLE 2 A user frequently presses “Ctrl+P” when he/she intends pressing “Ctrl+O”, so disables “Ctrl+P” as a shortcut for the Print function.

9.4 Pointing devices

9.4.1 General

The term “pointing device” in this section refers to any physical input 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, can use a finger tap or gesture in place of physical buttons, and these should be interpreted as equivalent to pointing-device button events and covered by provisions addressing such types of input.

NOTE ISO 9241-410 covers ergonomics factors for the design of mice, trackballs, touchscreens and touchpads, among other physical input devices.

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 the absolute position of the mouse pointer on the screen.

9.4.3 Provide easily-selectable pointing-device targets

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

NOTE This makes usage easier for all pointing device users and 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 [44].

NOTE It is desirable for applications to respect global OS settings for button reassignments 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, he/she 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 position to use for each function based on his/her 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), 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 can 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 The user can use the right click menu to achieve the same function as the double click.

EXAMPLE 2 Instead of holding the pointing-device button down to keep a pop-up open, the user can click to open it and click again to close it.

EXAMPLE 3 A user with a cognitive disability can single-click to perform a multi-click operation.

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 might have to be built into driver software.

NOTE 2 If the MouseKeys feature (see 9.1.2 and Annex E) is implemented fully in parallel with the standard pointing device, then this functionality would be provided since the keyboard could be used to hold down the pointing-device buttons. The provision is met if the platform provides this functionality and other software does not override it.

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 (allowing 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.

EXAMPLE 4 The application has a gesture where the user can erase numbers on-screen using a custom graphics tablet. The application provides the ability to have the graphics cursor puck buttons toggle on and off so the user does not have to keep the button held down manually while moving the cursor puck.

9.4.7 Provide adjustment of delay of pointing-device button-press acceptance

Software 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 1 s.

NOTE 1 A typical range for a adjustment of delay of pointing-device button-press acceptance feature is 0,1 s to 1,0 s.

NOTE 2 In most cases, this feature is supported automatically when software uses the standard pointing device input services provided by the platform software (and does not override this feature of the services).

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

9.4.8 Provide adjustment of minimum drag distance

Software should enable users to adjust the minimum pointer movement while the pointing-device button is held down, such that this will be registered as a drag event.

NOTE In most cases, this feature is supported automatically when software uses the standard pointing device input services provided by the platform software (and does not override this feature of the services).

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 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 [44].

NOTE In most cases, this feature is supported automatically when software uses the standard pointing device input services provided by the platform software (and does not override this feature of the services).

EXAMPLE 1 Users with slow movements might take a second or more between clicks of a double-click intended to open a document. Because they can adjust the time interval allowed between the two clicks, they can successfully double click.

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 inadvertent movement between clicks.

9.4.10 Provide adjustment of pointer speed

Software shall enable users to adjust the speed or ratio at which the pointer moves in response to a movement of the pointing device [44].

NOTE In most cases, this feature is supported automatically when software uses the standard pointing device input services provided by the platform software (and does not override this feature of the services).

EXAMPLE Users can 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 pointing device acceleration, it shall provide adjustment of the pointer movement acceleration including a setting of zero.

NOTE A zero acceleration setting allows assistive technology to move the pointer continuously with predictable results.

9.4.12 Provide adjustment of pointer movement direction

Software 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, opposite or perpendicular to the pointing movement direction.

NOTE 2 This is useful for people with movement limitations.

NOTE 3 In most cases, this feature is supported automatically when software uses the standard pointing device input services provided by the platform software (and does not override this feature of the services).

9.4.13 Provide a means of finding the pointer

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

EXAMPLE A user with low vision loses track of the mouse pointer. When the “Ctrl” 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 [44].

NOTE 1 The intent here is to replace or supplement concurrent actions with sequential input alternatives, because multiple simultaneous actions can be difficult or impossible for users with motor impairment.

NOTE 2 Most operating systems provide this function for all standard pointer buttons. Other software is generally responsible for implementing this feature only for pointer buttons used in combinations that are not covered by operating system features.

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 could induce seizures in individuals with photosensitive seizure disorders.

NOTE 1 Standards in this area are currently undergoing revision and adaptation to apply to new displays.

NOTE 2 Less than three flashes in any 1 s period meets all current standards.

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, except for simple progress indicators.

NOTE 1 A simple progress indicator has no movement other than to indicate current completion status.

EXAMPLE 1 A progress indicator consists of a status bar that shows completion along with an elf who is moving boxes. Clicking on the status indicator causes the elf to freeze but the status bar continues to reflect status.

NOTE 2 Individuals with low vision or reading problems need time to study information in order to comprehend it.

NOTE 3 Varying speed of presentation is also useful.

EXAMPLE 2 A Braille display is constantly refreshed to keep up with text output from the software. The user pauses the presentation so that he/she can read the Braille before it is refreshed.

EXAMPLE 3 A user presses the mouse button down on scrolling text, which pauses the moving text for as long as he/she holds the mouse button down, allowing the user to read the text.

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 audio 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 [44].

NOTE There are numerous cases where changing the view will necessarily change the meaning. The intent is that users have 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 will not affect the meaning.

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

EXAMPLE 4 A user changes the size of icons making it easier 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 by 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 is to enable the user to change the size of fonts and icons.

EXAMPLE 1 An operating system provides a “large print” setting that enlarges the fonts, lines and controls by a factor of 2 to make them easier to see. It also provides a magnifier to further enlarge portions of the screen.

EXAMPLE 2 An application allows font sizing and word wrap to allow documents to be enlarged up to a 72 point font size.

10.2.3 Use text characters as text, not as drawing elements

In graphical user interfaces, text characters should be used as text only, not to draw lines, boxes or other graphical symbols [44].

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.

NOTE 3 This does not refer to the use of characters within an image but only to the use of electronic text characters to create graphics (e.g. “ASCII ART”).

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 such graphics 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 *Virtual screen* is the name usually given to the viewing area extending the physical boundaries of the computer.

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 Do not convey information by visual font attribute alone

Software should not use visual font attributes as the only way to convey information or indicate an action.

EXAMPLE 1 Mandatory fields on a text entry form are indicated by bold text labels. An asterisk is added to the end of the mandatory field label so that this information is also available to blind users via speech output and to screen magnification users who cannot easily detect emboldening.

EXAMPLE 2 Menu items that are not active are indicated by “grey” or “dimmed” text. This status is also conveyed programmatically.

10.3.2 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 [1], [44], [49].

NOTE 1 If the platform software already provides this facility, the application can utilize it.

NOTE 2 This would apply despite the font size specified in a display document.

NOTE 3 The range of allowable sizes need not be unlimited. However, to be useful, it would need to include large font sizes.

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. The Web browser respects this setting and automatically enlarges any text that would otherwise be smaller than this size.

10.3.3 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 References [1], [44] and [49]).

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

NOTE 3 The range of allowable sizes need not be unlimited. However, to be useful, it would need to include large font sizes.

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 system’s 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 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. Its use 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 with disabilities

Software that includes colour schemes should provide colour schemes designed for use by people who have disabilities [45], [46], [47], [48], [49], [50], [51].

NOTE People who have visual disabilities, dyslexia, photophobia and sensitivity to screen flicker have colour preferences that affect their use of light-emitting displays.

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

10.4.3 Provide individualization of colour schemes

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

NOTE 1 The ability to share schemes is also useful.

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

EXAMPLE 1 A user adjusts the colour scheme provided for those with red-green colour blindness to optimize discriminability for his/her 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 colour scheme used to draw the different types of user-interface elements (such as windows, menus, alerts, keyboard focus cursors and default window background and text), the indicators of general states (such as keyboard focus and selection) and the codings for task-specific states (such as on-line, offline, or error).

10.4.4 Allow users to individualize colour coding

Except in cases where warnings or alerts have been standardized for mission-critical systems (e.g. red = network failure), software should allow users to individualize 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 representing links, an embedded application will 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.

NOTE Measures such as those proposed by the W3C [53] have been developed that provide contrast regardless of the colours used.

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 unique and 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 1 A user opens a second window of the same document using a word processing application. Both windows are of the same document and both are editable. The word processor adds a ":1" to the end of the document name to form the name of the first window. It names the second window with the same document name, except that it appends a ":2" to the end of the second window name so the two windows have meaningful yet unique names.

EXAMPLE 2 When a script or object hosted within a Web browser attempts to set the window's title to a string that is already used by another of the browser's windows, the browser modifies this string to be unique.

10.5.2 Provide window titles that are unique system-wide

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

NOTE This recommendation is specific to platforms because on many platforms an application cannot identify the windows belonging to other applications, as that would cause a security problem.

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 keyboard focus to any window currently running that is allowed to accept keyboard 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 keyboard focus.

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

10.5.4 Enable "always-on-top" windows

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

NOTE 1 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 keyboard focus from other windows (see 10.5.10).

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 he/she mouse-clicks on the on-screen keyboard another window keeps the keyboard 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 that manages windows 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 could wish to have multiple windows on top of everything else, for example, a calendar and clock. It can then 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 runs the magnifier on top so that it enlarges the on-screen keyboard.

10.5.6 Enable user choice of effect of pointer and keyboard focus on window stacking order

Software should allow users to choose to have the window that receives pointer or keyboard focus either automatically placed on top of all other windows (with the exception of an “always-on-top” window, see 10.5.4) 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.

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

Software shall provide a method for users to reposition all windows, including dialog boxes. Platform software that manages windows shall provide the user with an option to override any attempts by other software to prevent windows from being repositioned.

NOTE This helps, and can be required by, users working with several applications and/or windows, including assistive technology.

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

10.5.8 Enable window resizing

Software should provide a method for users to resize all windows, including dialog boxes. Platform software that manages windows should provide the user with an option to override any attempts by other software to prevent windows from being resized.

NOTE This provision is given as a recommendation and not as a requirement because as yet several widely-used operating systems do not provide dialog boxes that can be resized.

EXAMPLE A user with low vision uses a larger font size that causes text to run off the bottom of the dialog box. They enlarge the dialog box so they can see all of the text.

10.5.9 Support minimize, maximize, restore and close windows

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

NOTE 1 This helps the user to better use several applications and/or windows at the same time.

NOTE 2 This provision is given as a recommendation and not as a requirement because as yet several widely-used operating systems do not provide dialog boxes that can be resized.

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

NOTE When a window is designated to not accept the keyboard focus, any action that would normally be used to reassign the keyboard focus to that window will 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 keyboard 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 keyboard 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.

NOTE If software generates audio output, it is preferable that it provide its own control that adjusts the volume of its own audio output relative to other software and any system-wide volume setting.

EXAMPLE A user has a multimedia player application and a phone with an alert tone. He/she adjusts the first application’s volume control to a low setting and the second’s volume control to a high setting. The second now sounds louder than the first. When the user reduces system-wide volume setting in the operating system’s global preferences, the second application remains louder than the first, even though they are both reduced in volume.

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 [38].

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, 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 can 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 [38].

10.6.7 Allow users to choose visual alternative for audio output

If the hardware supports both audio and visual 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 [44], [46], [49], and software running on such systems shall support those options.

NOTE This feature is commonly called *ShowSounds* 5) (see Annex E).

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 Synchronize audio equivalents for visual events

Software shall synchronize 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 Audio is sometimes presented early or immediately after to avoid other audio events or real-time delays.

EXAMPLE A movie has audio descriptions of important visual information. The descriptions are timed to occur during gaps in the movie dialog.

10.6.9 Provide speech output services

If the hardware has the capability 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 users who are blind or have other reading disabilities and who depend on speech-based assistive technologies.

EXAMPLE SAPI (Speech API), Java Speech, Mac OS X TTS and GNOME speech 6).

5) ShowSounds™ is a trademark of the University of Wisconsin. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of the product named. Equivalent products may be used if they can be shown to lead to the same results.

6) SAPI, Java Speech, Mac OS X TTS and GNOME speech are examples of suitable products available commercially. This information is given for the convenience of users of this part of ISO 9241 and does not constitute an endorsement by ISO of these products.

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.

NOTE It is important for captions to be displayed in a way that provides sufficient contrast with their background (see 10.4.5).

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 the user to indicate that he/she wants available captions to be shown by all software.

NOTE A global setting to enable or disable captions is provided by ShowSounds, available on several major platforms (see Annex E).

10.7.3 Support system settings for captioning

Software that presents captions shall use system-wide caption preference settings by default. If the system-wide preference settings 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 resynchronize 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 them to minimize interference with visual content.

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

10.8 Media

10.8.1 Enable users to stop, start and pause

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

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

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

NOTE 1 “Replay” functions help users to avoid missing information.

NOTE 2 Following this recommendation is not always possible, especially in “real-time” presentations.

10.8.3 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 is able to see but not hear decides to view a captioned video with the audio turned off because he/she cannot determine the volume and do not want to disturb others.

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.

10.8.4 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, Reference [4]).

EXAMPLE When 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 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 independently 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 (Braille-code, Morse-code, etc.) will be mostly well experienced in tactile pattern of daily life.

EXAMPLE Bursts of tactile vibrations are designed to have a pattern similar to that of 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 On-line 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.

NOTE The use of technical terms is permitted where necessary for clearly explaining the functionality or product.

EXAMPLE The documentation of a computer-aided design (CAD) system can use terminology from the field of technical drawing.

11.1.2 Provide user documentation in accessible electronic form

All user documentation and "Help" shall be delivered in an electronic form that meets applicable documentation accessibility standards. This documentation shall be provided with the product, or upon request on a timely basis and without extra cost.

NOTE The category of "users" includes *administrators*. For software development software, "users" will include *software developers*.

11.1.3 Provide text alternatives in electronic documentation and "Help"

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 [44], [46], [49].

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 and for people who differ in terms of their preferred style of information processing (e.g. visual vs. verbal).

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 user's 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 in which operation of a specific device such as a mouse is required, a generic description might not be possible. However, such specific descriptions need only occur in "Help" in relation to using that device, not in all contexts.

EXAMPLE 1 The task description in "Help" does not require a user to recognize the colour of a user-interface element to use it, so the text does not state, for instance, "click on the green icon". Instead, 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 (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 easily discover the accessibility features of the software.

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 available 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 1 In countries where relay services are not provided free of charge, a company contracts with a relay services to assist the technical support process by providing real-time translation between the company's support staff and deaf customers who use text or video telephones to allow them to communicate in text or sign language. A similar service provides re-voicing for people whose speech is difficult to understand. The company also trains its technical support staff in how to optimize conversation through relay services.

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

EXAMPLE 3 Application Helpdesk operators are trained on the accessibility features so that they are able to guide a user through the process of carrying out a task or rectifying a fault entirely through the keyboard interface, without the need for any mouse click operations.

EXAMPLE 4 A company provides a dedicated telephone line for its customers who use telecommunications devices for the deaf, for example, V.18 [7)], and trains support staff on its use and etiquette so that users can communicate directly (rather than through relay) with customers support personnel.

EXAMPLE 5 An IVR (interactive voice response) system provides software support services that are accessible to text telephone users.

11.2.2 Provide accessible training material

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

7) Standard for the text telephone (TTY).

Annex A
(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. For the latest information on the series, see: http://isotc.iso.org/livelink/livelink?func=ll&objId=651393&objAction=browse&sort=name.

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 the 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)
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 9241-124)
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 technology (ICT) equipment and services	International Standard
Introduction		
100	Introduction to software ergonomics	Planned
General principles and framework		
110	Dialogue principles	International Standard
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 control principles	Planned
Presentation and support to users		
121	Presentation of information	Planned
122	Media selection and combination	Planned to revise and replace ISO 14915-3
123	Navigation	Planned to partially revise and replace ISO 14915-2
124	User guidance	Planned to revise and replace ISO 9241-13
129	Individualization	Planned
Dialogue techniques		
130	Selection and combination of dialogue techniques	Planned to incorporate and replace ISO 9241-1:1997/Amd 1:2001
131	Menu dialogues	Planned to replace ISO 9241-14
132	Command dialogues	Planned to replace ISO 9241-15
133	Direct-manipulation dialogues	Planned to replace ISO 9241-16
134	Form-based dialogues	Planned to replace ISO 9241-17
135	Natural language dialogues	Planned
Interface control components		
141	Controlling groups of information (including windows)	Planned to partially replace 9241-12
142	Lists	Planned
143	Media controls	Planned to partially revise and replace ISO 14915-2

Part no.	Subject/title	Current status
Domain-specific guidance		
151	Guidance on World Wide Web user interfaces	International Standard
152	Interpersonal communication	Planned
153	Virtual reality	Planned
Accessibility		
171	Guidance on software accessibility	International Standard
Human-centred design		
200	Introduction to human-centred design standards	Planned
210	Human-centred design of interactive systems	Planned to revise and replace ISO 13407
Process reference models		
220	Human-centred life cycle processes	Planned to revise and replace ISO/PAS 18152
Methods		
230	Human-centred design methods	Planned to revise and replace ISO/TR 16982
Ergonomic requirements and measurement techniques for electronic visual displays		
300	Introduction to electronic visual display requirements	To be published
302	Terminology for electronic visual displays	To be published
303	Requirements for electronic visual displays	To be published
304	User performance test methods	To be published
305	Optical laboratory test methods for electronic visual displays	To be published
306	Field assessment methods for electronic visual displays	To be published
307	Analysis and compliance test methods for electronic visual displays	To be published
308	Surface-conduction electron-emitter displays (SED)	To be published (Technical Report)
309	Organic light-emitting diode (OLED) displays	To be published (Technical Report)

Part no.	Subject/title	Current status
Physical input devices		
400	Principles and requirements for physical input devices	International Standard
410	Design criteria for physical input devices	International Standard
411	Laboratory test and evaluation methods for the design of physical input devices	Planned
420	Selection procedures for physical input devices	Under preparation
421	Workplace test and evaluation methods for the use of physical input devices	Planned
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
Application domains		
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

Annex B
(informative)

List of requirements

For the convenience of users of this part of ISO 9241, this annex identifies those subclauses giving requirements, which must be met in order to be able to claim conformance with this part of ISO 9241.

8.1.1 Provide a name for each user-interface element

8.1.4 Make names available to assistive technology

8.2.4 Enable individualization of the cursor and pointer

8.2.7 Enable user control of timed responses

8.3.1 Make controls for accessibility features discoverable and operable

8.3.3 Avoid interference with accessibility features

8.4.4 Provide alternatives when assistive technology is not available

8.4.5 Enable software-controlled media extraction

8.4.9 Allow warning or error information to persist

8.5.2 Enable communication between software and assistive technology

8.5.3 Use standard accessibility services

8.5.4 Make user-interface element information available to assistive technologies

8.5.5 Allow assistive technology to change keyboard focus and selection

8.5.6 Provide user-interface element descriptions

8.5.7 Make event notification available to assistive technologies

8.5.9 Use system-standard input/output

8.5.10 Enable appropriate presentation of tables

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

8.5.12 Allow assistive technology to monitor output operations

8.6.1 Read content on closed systems

8.6.2 Announce changes on closed systems

8.6.3 Operable through tactilely discernable controls

8.6.4 Pass through of system functions

9.1.2 Provide parallel keyboard control of pointer functions

9.2.1 Provide keyboard focus and text cursors

9.2.2 Provide high visibility keyboard focus and text cursors

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 accelerator 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 reassignment 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 to avoid 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 Synchronize audio equivalents for 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 Enable users to stop, start and pause
11.1.2 Provide user documentation in accessible electronic form
11.1.3 Provide text alternatives in electronic documentation and “Help”
11.1.5 Provide documentation and “Help” on accessibility features
11.2.1 Provide accessible support services

Annex C
(informative)

Sample procedure for assessing applicability and conformance

C.1 General

This annex provides an example of a checklist (see Table C.1) that can be used to determine whether the applicable requirements of this part of ISO 9241 have been met and its recommendations followed.

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

The checklist contains all requirements and recommendations from this part of ISO 9241, presented in sequence.

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

Use of the checklist provides a basis for

— determining which of the requirements and recommendations are applicable,

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

— providing a list in support of a claim of conformance showing that all applicable requirements have been met and all applicable recommendations followed.

The majority of the requirements are applicable to all software if it is intended 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 (users, tasks, environment and 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/subclause. 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 "Reason not applicable" column.

The next step involves determining whether the software being evaluated conforms to each requirement or recommendation (as 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 as well as observations on the method used or the judgment, which can be entered in the "Comments" column.

The completed checklist can be used in support of statements relating to conformance of software with this part of ISO 9241.

C.2 How to use the checklist

Clause numbers and titles are presented in the first two columns of Table C.1.

The third column is used to indicate whether the requirement or recommendation in each clause/subclause is applicable or not. All those that have no conditions attached to them already have a “Y” (for “Yes”) inserted in the third column to show that they are applicable, while “C” indicates applicability unless the specified conditions apply.

All other subclauses 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 indicated as being 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 (“Partially”) or not satisfied (“No”). Any clause/subclause 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 C.1 — Checklist for assessing applicability and conformity with this part of ISO 9241

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
8	**General guidelines and requirements**						
8.1	**Names and labels for user-interface elements**						
8.1.1	Provide a name for each user-interface element	Y					
8.1.2	Provide meaningful names						
8.1.3	Provide unique names within context						
8.1.4	Make names available to assistive technology	Y					
8.1.5	Display names						
8.1.6	Provide names and labels that are short						
8.1.7	Provide text label display option for icons						
8.1.8	Properly position the labels of user-interface elements on the screen						
8.2	**User preference settings**						
8.2.1	Enable individualization of user-preference settings						
8.2.2	Enable adjustment of attributes of common user-interface elements						
8.2.3	Enable individualization of the user interface look and feel						
8.2.4	Enable individualization of the cursor and pointer	C					
8.2.5	Provide user-preference profiles						
8.2.6	Provide capability to use preference settings across locations						
8.2.7	Enable user control of timed responses	C					

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
8.3	**Special considerations for accessibility adjustments**						
8.3.1	Make controls for accessibility features discoverable and operable	**Y**					
8.3.2	Safeguard against inadvertent activation or deactivation of accessibility features						
8.3.3	Avoid interference with accessibility features	**Y**					
8.3.4	Inform user of accessibility feature “On”/“Off” status						
8.3.5	Inform user of accessibility feature activation						
8.3.6	Enable persistent display						
8.4	**General control and operation guidelines**						
8.4.1	Enable switching of input/output alternatives						
8.4.2	Optimize the number of steps required for any task						
8.4.3	Provide “Undo” and/or “Confirm” functionality						
8.4.4	Provide alternatives when assistive technology is not operable	**C**					
8.4.5	Enable software-controlled media extraction	**C**					
8.4.6	Support “Copy” and “Paste” operations						
8.4.7	Support “Copy” operations in non-editable text						
8.4.8	Enable selection of elements as an alternative to typing						
8.4.9	Allow warning or error information to persist	**Y**					
8.4.10	Present user notification using consistent presentation techniques						
8.4.11	Provide understandable user notifications						
8.4.12	Facilitate navigation to the location of errors						

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
8.5	**Compatibility with assistive technology**						
8.5.1	General						
8.5.2	Enable communication between software and assistive technology	Y					
8.5.3	Use standard accessibility services	Y					
8.5.4	Make user-interface element information available to assistive technologies	Y					
8.5.5	Allow assistive technology to change keyboard focus and selection	Y					
8.5.6	Provide user-interface element descriptions	C					
8.5.7	Make event notification available to assistive technologies	Y					
8.5.8	Allow assistive technology to access resources						
8.5.9	Use system-standard input/output	C					
8.5.10	Enable appropriate presentation of tables	Y					
8.5.11	Accept the installation of keyboard and/or pointing device emulators	Y					
8.6.12	Allow assistive technologies to monitor output operations	Y					
8.5.13	Support combinations of assistive technologies						
8.6	**Closed systems**						
8.6.1	Read content on closed systems	Y					
8.6.2	Announce changes on closed systems	Y					
8.6.3	Operable through tactilely discernable controls	Y					
8.6.4	Pass through of system functions	Y					

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	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						
9.1.5	Provide system-wide spell-checking tools						
9.2	**Keyboard focus**						
9.2.1	Provide keyboard focus and text cursor	Y					
9.2.2	Provide high visibility keyboard focus and text cursors	Y					
9.2.3	Restore state when regaining keyboard focus						
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					
9.3.5	Provide adjustment of same-key double-strike acceptance	Y					
9.3.6	Provide adjustment of key repeat rate						
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						

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
9.3.11	Provide implicit or explicit designators						
9.3.12	Reserve accessibility accelerator key assignments	Y					
9.3.13	Enable remapping of keyboard functions						
9.3.14	Separate keyboard navigation and activation	Y					
9.3.15	Follow platform keyboard conventions						
9.3.16	Facilitate list and menu navigation						
9.3.17	Facilitate navigation of controls by grouping						
9.3.18	Arrange controls in task-appropriate navigation order						
9.3.19	Allow users to customize accelerator keys						
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						
9.4.4	Enable the reassignment 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					
9.4.7	Provide adjustment of delay of pointing-device 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					

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
9.4.11	Provide adjustment of pointer 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					
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	Provide keyboard access to information displayed outside the physical screen	C					
10.3	**Text/fonts**						
10.3.1	Do not convey information by visual font attribute alone						
10.3.2	Enable users to set minimum font size						
10.3.3	Adjust the scale and layout of user-interface elements as font-size changes						

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
10.4	**Colour**						
10.4.1	Do not convey information by colour output alone	Y					
10.4.2	Provide colour schemes designed for people with disabilities						
10.4.3	Provide individualization of colour schemes						
10.4.4	Allow users to individualize colour coding						
10.4.5	Provide contrast between foreground and background						
10.5	**Window appearance and behaviour**						
10.5.1	Provide unique and meaningful window titles						
10.5.2	Provide window titles that are unique system-wide						
10.5.3	Enable non-pointer navigation 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 pointer and keyboard focus on window stacking order						
10.5.7	Enable window positioning	Y					
10.5.8	Enable window resizing						
10.5.9	Support minimize, maximize, restore and close windows						
10.5.10	Enable windows to avoid taking focus	Y					

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
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						
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	Synchronize audio equivalents for 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**						
10.8.1	Enable users to stop, start and pause	Y					
10.8.2	Enable users to replay, rewind, pause and fast- or jump-forward						
10.8.3	Allow user to control presentation of multiple media streams						
10.8.4	Update equivalent alternatives for media when the media changes						

Table C.1 (*continued*)

Clause/subclause of this part of ISO 9241		Applicability		Conformance			
		Yes/No	Reason not applicable	Yes	Partially	No	Comments
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	**On-line documentation, "Help" and support services**						
11.1	**Documentation and "Help"**						
11.1.1	Provide understandable documentation and "Help"						
11.1.2	Provide user documentation in accessible electronic form	**Y**					
11.1.3	Provide text alternatives in electronic documentation and "Help"	**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 D
(informative)

Activity limitation issues

D.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 by the ICF classification [55], the same limitations can arise from other sources, such as the particular context in which individuals find themselves at a given moment.

For the purposes of this annex, three main area of the ICF classification are considered most relevant to interaction with software systems:

a) sensory functions, including seeing and hearing;

b) neuromusculoskeletal and movement related functions;

c) 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.

D.2 Sensory functions

D.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.

D.2.1.1 Individuals who are unable to see

A user who is unable to see will have to use his or her remaining senses and appropriate provision will have to be made to enable access to equivalent content and resources via those senses. In addition, individuals could have normal vision but be unable to view a screen due to context or task-related issues — for example, while 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, joystick, voice or other control actuator.

Some individuals who cannot see will use specialized assistive technologies. For example, individuals who have learned Braille can take advantage of software and hardware that will provide *screen readers* that will produce Braille output. Those who become blind later in life are less likely to learn such specialized skills, although they might learn some new auditory skills and thus rely on additional auditory methods of obtaining information.

Screen readers — 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 mainly on speech to convey information. Others use tactile displays such as dynamic Braille or Moon [8)] 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) provided visually be available to assistive technologies for alternative display.

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

D.2.1.2 Individuals with low vision

Persons with low vision often use technologies more commonly associated with those who are unable to see at all (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 those with low vision should not be placed in the same situation 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 can 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 can experience particular difficulties in detecting size coding, 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 individualization. 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 can cause difficulties when very small displays, such as those on printers, copiers and ticket machines, are required to be read.

D.2.2 Hearing

Auditory feedback (both speech and non-speech) and automatic speech recognition of user input have become increasingly important elements of software interaction.

---

8) Moon text is a tactile script for the blind composed of curved lines.

D.2.2.1 Individuals who are unable to hear

Individuals who are unable to hear sound are thus 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 can 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 might 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 (which 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. on a machine shop floor) or where sound is turned off or cannot be used (e.g. in a library).

Some individuals with a general inability to detect auditory information can also experience limitations on voice and speech functions. This can have implications for their ability to produce speech recognizable 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 deaf), 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.

D.2.2.2 Individuals with a reduced ability to hear

Individuals can 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 can arise include

— the inability to detect sound,

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

— difficulty in localizing sounds,

— difficulty in discerning sounds against background noise, and

— 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. In addition, individuals with a reduced ability to hear can adjust auditory material through the ability to individualize the characteristics of auditory presentations (e.g. increasing volume or selectively changing the frequency spectrum used).

Individuals with a reduced ability to hear might or might 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).