Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 16 March 2015

Case Number: T 0659/11 - 3.5.06

Application Number: 03729009.5

Publication Number: 1546857

IPC: G06F9/44

Language of the proceedings: EN

Title of invention:
SYSTEM AND METHOD FOR MAKING USER INTERFACE ELEMENTS KNOWN TO AN APPLICATION AND USER

Applicant:
Microsoft Technology Licensing, LLC

Headword:
Control pattern/MICROSOFT

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step - (no)

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern
Boards of Appeal
Chambres de recours

European Patent Office
D-80298 MUNICH
GERMANY
Tel. +49 (0) 89 2399-0
Fax +49 (0) 89 2399-4465

Case Number: T 0659/11 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 16 March 2015

Appellant: (Applicant)	Microsoft Technology Licensing, LLC One Microsoft Way Redmond, WA 98052 (US)
Representative:	Grünecker Patent- und Rechtsanwälte PartG mbB Leopoldstraße 4 80802 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 19 October 2010 refusing European patent application No. 03729009.5 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Sekretaruk
Members:	S. Krischer
	G. Zucka

Summary of Facts and Submissions

I. The appeal is directed against the decision of the examining division, posted on 19 October 2010, to refuse the application 03729009. The reasons for the refusal are for the main request lack of original disclosure (Article 123(2) EPC) and lack of inventive step (Article 56 EPC 1973) over the following document, and for the remaining requests lack of inventive step only:

D1 Anonymous: "GNOME Accessibility for Developers - DRAFT: How to make GNOME 2.0 Applications Accessible, Examples that Use the Accessibility API", Internet article, August 2002, pages 1-6, XP2443743, retrieved on 23 July 2007 from http://web.archive.org/web/20020821064040/developer.gnome.org/projects/gap/guide/gad/gad-api-examples.html.

The following document is also cited in the appealed decision, but not used in its argumentation:

D2 Anonymous: "ATK Implementation Proposal Draft 0.5", Internet article, 2 November 2001, pages 1-24, XP2443745, retrieved on 23 July 2007 from http://web.archive.org/web/ 20011102020337/ http ://developer.gnome.org/projects/gap/tech-docs/GTKimpl.html.

II. A notice of appeal was received on 15 December 2010. The appeal fee was received the same day. A statement of the grounds of appeal was received on 16 February 2011. The appellant made a main and two auxiliary requests and requested oral proceedings.

III. In its summons to oral proceedings, the board gave reasons for its preliminary opinion that all of the requests lacked an inventive step over D1. The summons was silent with respect to Article 123(2) EPC, since the main request of the refusal decision was not maintained.

IV. In a letter dated 23 February 2015, the appellant filed claim sets of new auxiliary requests 1 and 2.

V. Oral proceedings were held on 16 March 2015 during which the appellant filed the following document:

Anonymous: "AtkACTION: ATK - Accessibility Toolkit", Internet article, pages 1-4, retrieved on 13 March 2015 from https://developer.gnome.org/atk/unstable/AtkACTION.html.

At the end of the oral proceedings, the board announced its decision.

VI. The appellant requests that the decision be set aside and a patent be granted on the basis of claims 1-15 of a main request (the refused first auxiliary request) filed on 24 August 2010, claims 1-15 of a first auxiliary request or claims 1-13 of a second auxiliary request, both filed on 23 February 2015.
The further text is: description pages 2, 4-14 as published, pages 1, 3, 3b, 15 filed on 26 May 2008, page 3a filed on 10 November 2008; drawing sheets 1-6 as published.

VII. Claim 1 of the main request reads as follows:

"1. A computer-implemented method for programmatically manipulating a user interface element of an application (202), the method comprising:

gathering (513), by an automation utility (201), from the user interface element of the application (202) information about the user interface element, the information being stored within a property of the user interface element;

conveying, by the automation utility (201), the information to the user;

requesting (515), by the automation utility (201), from the user interface element of the application (202) whether the user interface element supports a control pattern, the user interface element being of a particular user interface element type, the control pattern describing basic functionality exposed by a plurality of types of user interface elements; and

if the user interface element supports the control pattern,

returning an interface that comprises at least one method supported by the user interface element, and

programmatically manipulating (517), by the automation utility (201), the user interface element using the at least one method comprised in the returned interface and exposed by the user interface element, the returned interface corresponds to the control pattern,

whereby the user interface element is programmatically manipulated by the automation utility through the returned interface associated with the control pattern

based on its support of the control pattern without reference to the user interface element's type."

VIII. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that the third and subsequent steps read (additions marked in *italics*):

"requesting (515), by the automation utility (201), from the user interface element of the application (202) whether the user interface element supports a control pattern, the user interface element being of a particular user interface element type, the control pattern *being associated with at least one predefined method included in one or more user interface elements, the control pattern* describing basic functionality exposed by a plurality of types of user interface elements; and

if the user interface element supports the control pattern,

returning an interface that comprises *the* at least one method *for the control pattern that is* supported by the user interface element, and

programmatically manipulating (517), by the automation utility (201), the user interface element *by* using the at least one method comprised in the returned interface and exposed by the user interface element, the returned interface corresponds to the control pattern,

whereby the user interface element is programmatically manipulated by the automation utility through the returned interface associated with the control pattern based on its support of the control pattern without reference to the user interface element's type."

IX. Claim 1 of the second auxiliary request differs from claim 1 of the first auxiliary request in that the third and the subsequent steps read (additions marked in *italics*; deletions are ~~struck through~~):

"requesting (515), by the automation utility (201), from the user interface element of the application (202) whether the user interface element supports a *plurality of* control pattern*s*, the user interface element being of a particular user interface element type, *each of* the *plurality of* control pattern*s* being associated with at least one predefined method included in one or more user interface elements, *each of* the *plurality of* control pattern*s* describing basic functionality exposed by a plurality of types of user interface elements, *wherein the user interface element supports a plurality of different control patterns*; and

if the user interface element supports the control pattern,

returning an interface that comprises the ~~at least one~~ *plurality of* method*s* for the control pattern ~~that is~~ supported by the user interface element, and

programmatically manipulating (517), by the automation utility (201), the user interface element by using at least one *of the plurality of* method*s* comprised in the returned interface and exposed by the user interface element~~, the returned interface corresponds to the control pattern~~,

whereby the user interface element is programmatically manipulated by the automation utility through the

returned interface associated with the *plurality of* control pattern*s* based on its support of the *plurality of* control pattern*s* without reference to the user interface element's type."

Reasons for the Decision

1. *Overview of the invention*

The application relates to programmatically manipulating user interface (UI) elements of an application program by a so-called "automation utility", such as an assistive technology application (e.g. a screen reader which narrates UI elements via a text-to-speech engine to visually impaired persons or which sends such information to a Braille display; original description page 1, lines 20-23, 14-15, 28), an automated testing script, a macro recorder, a commanding application (page 4, lines 27-29; original claim 5), a speech and dictation software or a "command and control utility" (page 7, lines 27-30; original claim 5). Programmatically manipulating a UI element means for example that the automation utility "presses" a UI button by calling a method (page 9, last but one box at the bottom of the table: "Invoke - ... For example, a button supports this pattern to allow an automation utility to programmatically *press* the button.").

The claimed computer-implemented method is two-fold: The first part consists of the first two steps (which stem from original claim 7). The second part consists of the remaining steps.

In the first part, the automation utility requests information (called "properties"; e.g. the name of a button, page 12, lines 1-3) from the UI element and "conveys" this information to the user (e.g. narrates the button name "Help Button" to the user; page 12, first paragraph; see also page 13, second paragraph and figure 5 (513)). According to the description, this may for example happen in the context of the user navigating to a UI element, e.g. a button (page 11, lines 17-22; page 12, line 1).

In the second part, the automation utility queries whether the UI element supports a specific "control pattern", like the "Invoke control pattern" (page 12, lines 12-14; page 14, lines 12-13; figure 6 (605)). If yes, the automation utility receives an "interface" with a method name (e.g. method name "Y" of an interface named "X" in figure 4, (414)) from the UI element (page 12, lines 14-16; page 13, paragraph 4). The automation utility then manipulates the UI element with the help of the method [name] from the received interface (e.g. presses the button for invoking the button function by calling the method "Y"; page 12, lines 14-16; page 13, lines 19-21 and figure 5 (517); page 14, lines 30-33).

2. *Overview of the decision*

Claim 1 of all requests lacks an inventive step (Article 56 EPC 1973).

3. *Inventiveness*

3.1 Main request

3.1.1 The main request is identical to the refused first auxiliary request. In decision section 4, claim 1 was found to lack inventive step over D1, also referring back to section 3 (in particular to 3.3-3.6 for the second part of the claim from which differentiating feature (III) stems).

3.1.2 The decision (3.3) identifies the interface names of D1 (e.g. "ACTION" or "COMPONENT" on page 2, lines 17-24; or "TEXT" on page 4, lines 9-11) with the names of the "control pattern" in the claim (see also description pages 9-10, the left column of table 1: "Invoke", "Sort" or "Window"). This identification of the decision is also mentioned in the grounds for refusal (page 2, paragraph 4) and was not contested by the appellant until the oral proceedings, during which the appellant argued that in addition to differentiating feature (III) (i.e. returning an interface instead of a boolean value as in D1), a control pattern was different from an interface and was missing in D1.

3.1.3 According to the board's view as set out during oral proceedings, the control pattern in the application is an *informal notion* (see pages 9-10, table 1 which merely discloses names and natural language descriptions). It is formally represented by the (one) interface corresponding to the respective control pattern. A program (like the automation utility of the claim) cannot deal with informal notions. Therefore, the third step in the claim, which requests whether the UI element supports a control pattern, is considered to be a request whether the UI element implements the *interface* corresponding to the control pattern in question or, more precisely, whether the UI element

implements the *methods* of the interface of the control pattern in question (see page 13, paragraph 4).

3.1.4 It has to be noted that in this context the word "interface" does not relate to "user interface", but to "application programming interface" (API). Throughout the whole description, the application uses the expression "*user interface*" to designate a graphical user interface (GUI) of an application (202 in figure 2; page 6, last paragraph) and "user interface element" for a GUI control of that application (see page 3, last paragraph, first sentence; e.g. a button). As to the word "*interface*" (without the word "user"), its most concrete disclosure in the application is on page 12, second paragraph which refers to figure 4: item 414 is the only example of an interface in the application and reads:

"I'face X{
Method Y
Method Z}"

This is what is usually called an API (except that the parameter types and the result types usually also contained in an API are missing). Technically, it results in a collection of method *names* (here "Y" and "Z") which have to be implemented by properly programmed *methods* (with the same names) in a program (or in a program library). D1 also relates to an API, see the title of D1: "Examples that Use the Accessibility API".

3.1.5 It seems that these two meanings of the word "interface" were mixed up in the appellant's letter of reply dated 23 February 2015 (after the summons for

oral proceedings). It is stated on page 2, paragraph 2, sentence 5 that:

> "In contrast, D1 merely returns an indication of the existence of an *interface element* in response to a query for that particular *interface element*." (emphasis added)

However, D1 is not about (user) interface elements (which are controls of the GUI toolkit *GTK*), but about the accessibility API for GTK, called *ATK*. According to D1, page 2, lines 18-24, the macro function ATK_IS_ACTION returns TRUE if the concerned UI element implements the interface "ACTION" (which allows for example to click a button, see D2, page 8, line 13).

3.1.6 Therefore, the board is not convinced by the appellant's argument that the control pattern is a further differentiating feature, but it agrees with the decision that the request in D1 whether a UI element implements an ATK interface can be identified with the request in the claim whether a UI element supports a control pattern.

3.1.7 It follows that the only feature in which the claim differs from D1 is feature (III) of the decision. The grounds (page 3, paragraphs 1-4) mainly dispute that it would be obvious to arrive at feature (III) starting from D1 (as stated in decision sections 3.3-3.5). In other words, the grounds do not consider it to be obvious that the application (202) returns the requested interface of a supported control pattern when starting from D1 which merely returns TRUE or FALSE on the request about the support of an interface.

3.1.8 The first argument of the grounds (page 3, first paragraph) is that there is no hint in D1 that the skilled person would have done so.

3.1.9 However, D1 apparently assumes that the automation utility already has access to the interface of interest. The board further expects that also in the application the automation utility program already has access to the interface before it is returned: assume that the returned interface of the control pattern "Window" from table 1 on page 10 contains method names "a" to "e" to change the position and the size of a window, to maximise and minimise it and to make it fullscreen. How should the automation utility program (or its programmer) know which one of the methods "a" to "e" of the returned interface does what in order to call the appropriate method to apply the intended functionality to the UI element?

3.1.10 Be that as it may, since D1 is a description of a certain implementation, it is not surprising that it does not describe an implementation variant which the system described in D1 does not provide.

3.1.11 Furthermore, the board has a similar opinion as the decision (3.3): The board considers it to be an obvious design option to change the method from returning boolean values (TRUE, FALSE) to returning an interface (as disclosed in figure 4 (414)). Thus, there is no need to disclose such an implementation detail in D1. The skilled person would consider it obvious to change the method of D1 accordingly if the program context required it.

3.1.12 Moreover, the application does not disclose *how* an interface is to be returned in the application. The board can only speculate whether this might be done by a reference (i.e. by a name, an ID or a pointer) or by the content of the interface (i.e. the method names).

3.1.13 In their second argument, the grounds (page 3, paragraphs 2-4) repeat an argument from the letter of 24 August 2010 (page 4, paragraphs 3-6) that the methods of the interface had the sole purpose of retrieving information from a UI element, whereas the invention only did so in the first part of the claim when retrieving property information. In the second part of the claim, the methods of the returned interface only manipulated the UI element.

3.1.14 The board cannot accept these lines of argument. Firstly, D1 also discloses methods which manipulate UI elements, as stated in the decision (last paragraph of 3.4) with respect to the interface ACTION (D1, page 2, line 21). The board notes that according to D2 (page 8, line 13) the interface ACTION manipulates a button by clicking, pressing and releasing it.
Secondly, the description of the application also discloses, among others, interface methods which do not manipulate the UI object, but retrieve information from it, see page 12, line 16: "In another example, a Selection control pattern (associated with the combo box 320) may provide methods to query for selected items".

3.1.15 Thus, the board takes the view that the interface methods of D1 correspond to the methods of the returned interface of the claim (cf. decision 3.4 and grounds, page 3, paragraph 5).

3.1.16 In their third argument, the grounds (page 4, paragraph 3) state that an automation utility using the method of D1 "additionally need[s] to obtain the capability of manipulating that user interface element", either by already having the "capability" or by retrieving/obtaining a method for manipulating.

3.1.17 From a technical point of view it seems to be undeniable that the technique of D1 gives to an automation utility the "capability" of calling the methods of an interface like ACTION, COMPONENT or TEXT. It is assumed that in D1 as well as in the invention this happens in the way which usual programming languages allow (e.g. by linking libraries to the executable code of the automation utility), since neither D1 nor the application disclose any specific technique for this.

3.1.18 Therefore, claim 1 of this request is considered not to be inventive in the sense of Article 56 EPC 1973.

3.2 *First auxiliary request*

3.2.1 It was not disputed by the appellant during oral proceedings that claim 1 of the first auxiliary request merely clarifies claim 1 of the main request.

It follows that the same reasoning as for the main request also applies to the first auxiliary request.

3.2.2 Therefore, claim 1 of this request is considered not to be inventive in the sense of Article 56 EPC 1973.

3.3 *Second auxiliary request*

3.3.1 Claim 1 of this request differs from claim 1 of the main request by specifying that the method of the main request also works for a *plurality* of control patterns.

3.3.2 However, also the UI elements of D1 can implement a plurality of ATK interfaces, see D2 (which describes further details of the framework of D1), page 9, section "GtkButton Notes" where the UI element GtkButton implements the ATK interfaces ACTION, COMPONENT, TEXT etc. It is furthermore obvious for a skilled person who wants to program a request whether a UI element supports a plurality of control patterns to call the corresponding ATK_IS_... macro functions in its automation utility program one after another.

3.3.3 Therefore, claim 1 of this request is considered not to be inventive in the sense of Article 56 EPC 1973.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

B. Atienza Vivancos W. Sekretaruk

Decision electronically authenticated