Internal distribution code:

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

Datasheet for the decision of 21 February 2014

Case Number: T 2205/10 - 3.5.06

Application Number: 03767540.2

Publication Number: 1563370

IPC: G06F9/44

Language of the proceedings: EN

Title of invention:
MODELING SYSTEM FOR GRAPHIC USER INTERFACE

Applicant:
SAP AG

Headword:
GUI modeling/SAP

Relevant legal provisions:
EPC Art. 83

Keyword:
Sufficiency of disclosure - (no)

Decisions cited:

Catchword:

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 2205/10 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 21 February 2014

Appellant: (Applicant)	SAP AG Dietmar-Hopp-Allee 16 69190 Walldorf (DE)
Representative:	Müller-Boré & Partner Patentanwälte PartG mbB Friedenheimer Brücke 21 80639 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 30 June 2010 refusing European patent application No. 03767540.2 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman: D. Rees
Members: S. Krischer
M.-B. Tardo-Dino

Summary of Facts and Submissions

I. The appeal is directed against the decision of the examining division, posted on 30 June 2010, to refuse the application 03767540 for lack of inventive step of the main request and for lack of clarity of the auxiliary request. The following document was cited in relation to inventiveness:

D1 US 2001/45963 A1, 29 November 2001

In an obiter dictum of the decision, it was objected that there also was a lack of inventive step of the auxiliary request.

II. A notice of appeal was received on 18 August 2010. The fee was received on 19 August 2010. A statement of the grounds of appeal was received on 13 October 2010. Claim sets of a main and an auxiliary request were filed. Oral proceedings were requested.

III. In its summons to oral proceedings, the board gave reasons for its preliminary opinion that the application suffered from a lack of clarity of its claims (Article 84 EPC) and a lack of disclosure as to how the invention claimed is to be carried out (Article 83 EPC). Furthermore, the board was provisionally of the opinion that claim 1 of the two requests lacked an inventive step (Article 56 EPC).

IV. In a letter dated 21 January 2014, the appellant filed a claim set of a second auxiliary request.

V. Oral proceedings were held on 21 February 2014. At their end, the board announced its decision.

VI. The appellant requests that that the decision under appeal be set aside and that a patent be granted on the basis of claims 1-10 of the main request (corresponding to the main request of the appealed decision) or of the first auxiliary request (slightly differing from the auxiliary request refused by the examining division), both filed with the grounds of appeal, or of the third auxiliary request filed on 21 January 2014, description pages 2, 34 as filed on 18 June 2007, pages 1, 3-33, 35-39 as originally filed, (corrected) drawing sheets 1-23 as published on 24 March 2005.

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

"1. A computer-implemented method for generating a user interface (220), the user interface (220) being configured for use in a client-server environment, the method comprising:

providing an editor (311) for designing a visual representation (310) of a user interface model, the visual representation (310) being a drawing that specifies functions the user interface (220) is supposed to perform,

the editor (311) providing a workspace (804) and a task panel (808) to be displayed on a display device on a client system, the workspace (804) being provided to design the visual representation (310) thereon, the task panel (808) providing a plurality of elements for use in designing the visual representation (310), one or more of the elements being associated with a server system remotely located from the client system; and

the editor (311) translating the visual representation (310) to a machine-readable canonical representation (320), and then

translating the canonical representation (320) into user interface code (330)."

VIII. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that the last two paragraphs of claim 1 of the main request are replaced by (differences marked in *italics*):

"the editor (311) translating the visual representation (310) to a machine-readable canonical representation (320) *in the form of GUIMachine Language (GML)*, and then

translating the canonical representation (320) into user interface code (330),

wherein the canonical representation (320) enables providing a structured user interface modelling system repository (321) for the user interface model, the repository serving as a source for code generation tools and producing a semantically rich source of user interface knowledge that is exploitable by secondary tools, and

wherein the canonical representation includes three main layers comprising an information layer (701) defining the information objects that can be received or sent to an underlying back-end application and the functions that can be called, an interaction model layer (703) defining the types of users that are expected to use the user interface, the tasks that they are supposed to accomplish by using the user interface, and the specific user interface dialogues that are required for carrying out each task, and a presentation model layer (705) defining how a user interface will appear, including topology, geometry, and styling."

IX. Claim 1 of the second auxiliary request reads as follows (wording added to the main request marked in *italics*; deletions are ~~struck through~~):

"1. A computer-implemented method for generating a user interface (220), the user interface (220) being configured for use in a client-server environment, the method comprising:

providing an editor (311) for designing a visual representation (310) of a user interface model,

- the visual representation (310) being a drawing that specifies functions the user interface (220) is supposed to perform,

- the editor (311) providing a workspace (804) and a task panel (808) to be displayed on a display device on a client system,

- the workspace (804) being provided to design the visual representation (310) thereon,

- the task panel (808) providing a plurality of elements for use in designing the visual representation (310), one or more of the elements being associated with a server system remotely located from the client system;

creating the visual representation (310) using the editor (311); and

~~the editor (311)~~ translating, *by the editor (311)*, the visual representation (310) *in*to a*n XML-based* ~~machine-readable~~ canonical representation (320), and then

translating the canonical representation (320) into *executable* user interface code (330)."

X. Each request also includes a corresponding independent system and "computer readable medium" claim.

Reasons for the Decision

1. *Overview*

1.1 The application relates to generating a graphical user interface (GUI) by a user with the help of an editor. The user enters a visual representation of the GUI to be modelled by using the GUI of the editor ([70] of the original description). A visual representation is - according to the description - a tree of nested diagrams ([97]) or a drawing specifying the functions of the GUI ([72]; claim 1 of all requests). See figures 10A and 11A for two examples of visual representations. The editor translates the visual representation to an "XML-based canonical representation" ([74]; claim 1 of the second auxiliary request). Examples for canonical representations are found in figures 10B and 11B, corresponding to appendices A and B of the description. Then, the canonical representation is translated to executable GUI code ([70], [77], [168], [169]; figure 3B; claim 1 of the second auxiliary request).

1.2 The application suffers from a lack of disclosure as to how the invention claimed is to be carried out (Article 83 EPC).

2. *Admissibility of the second auxiliary request*

Since this request clearly represents an attempt to overcome the clarity objections (Article 84 EPC) noted in the summons (4.2, 4.4) and does not raise any new issues, it is admitted to the procedure.

3. *Insufficient disclosure*

3.1 During the oral proceedings before the board, insufficient disclosure was discussed first, being the most fundamental objection. Once the board came to the conclusion that the disclosure was indeed insufficient, discussions about clarity and inventive step were superfluous. Since these issues were not dealt with in the oral proceedings, they will likewise not be considered in what follows.

3.2 After having discussed at length the sufficiency of the disclosure with respect to claim 1 of the *second* auxiliary request, the board stated that it considered this claim to be the most concrete one with respect to the canonical representation (with the characterisation "XML-based"; without the unclear expression "machine-readable") compared with claim 1 of the main and the first auxiliary request. Therefore, if the invention as claimed in the second auxiliary request were insufficiently disclosed, then this would also be the case for the main request and the first auxiliary request.
The appellant accepted that this was the situation.

3.3 Thus, the following objections and arguments relate to claim 1 of the second auxiliary request. They concern the three main elements of the claim:

- the visual representation
- the XML-based canonical representation
- the two translations: visual representation -> canonical representation -> executable GUI code

3.4 According to the appellant during the oral proceedings, the *visual representation* is generated by dragging and dropping symbols from the task panel (the right box in figure 11A) to the workspace (the big box at the left). The available symbols are shown in figure 4 (see [100]). This defines and constrains the possible visual representations. For example, in figure 11A the boxes "Bank Getlist" (1106) and "Bank Getdetail" (1110) have been added and in-/out-connected with other added boxes, e.g. with the box "Search Form" (1104) (see also [171]). The editor has predefined XML code portions for each symbol of figure 4. When a symbol is dragged from the task panel and dropped into the workspace, the corresponding XML code portion is inserted into the canonical representation. For example, in appendix B of the description which shows the canonical representation of figure 11A, one can find the strings "Bank Getlist" (middle of page 38) and "Bank Getdetail" (bottom of page 37) together with their parents, position, size and in-/out-connections. Then, the canonical representation is translated to executable GUI code according to the skilled person's general knowledge about translating and generating GUI code.
The core of the invention is to create visually the visual representation, to translate it into the canonical representation and then into executable GUI code for possibly several platforms.

3.5 The board is not convinced by these explanations. Paragraph [100] does not state that figure 4 discloses an exhaustive set of the symbols which are the primitives to build the formal language of the visual representations and, indeed, indicates that other "symbolic vocabularies" may be used. In fact, the

symbols used for "Bank Getlist" (1106) and "Bank Getdetail" (1110), i.e. boxes with convex curves as the top and the bottom edges, are not contained in the symbol set of figure 4. On the other hand, the application also does not disclose what is the minimal set of necessary symbols to form the visual representation language.

3.6 According to the appellant, this is left to the skilled person: If he does not want to have list boxes in his implementation, he does not use them. Further, the skilled person could decide to implement a *very simple GUI*, e.g. only creating a window with an "OK"-button and an "exit"-button. At least this would be, according to the appellant, sufficiently disclosed.

3.7 Firstly however the application indicates that the symbols given in figure 4, and what they represent, are not necessarily meant to be a matter of common general knowledge, for example "scenario", "scene" or "actor". And the board judges that after having read the description of them in the application (e.g. [108], [118], [123]), the skilled person would still not know what a GUI to be generated for them should look like. As to the second argument, the generally known GUI widgets such as buttons, menu bars, text display boxes or text entry fields simply do *not* appear in figure 4. Therefore, a skilled person is not even able to implement the claimed method restricted to very simple GUIs without departing from what *is* disclosed.

3.8 The application does not describe a mere GUI builder which allows a designer to compose the aforementioned generally known GUI widgets in one or more windows. Instead the application introduces high-level, abstract

metaphors or concepts like scenario, scene, actor, bag or cluster (figure 4), but fails to sufficiently disclose their technical meaning, technical implications and their implementation. What is missing is the nature of the GUI elements which should be generated for each of the symbols in figure 4. Paragraphs [100]-[136] describing the symbols of figure 4 do not deliver this information. And among the figures, there is only one single example of a generated GUI, presented almost identically on three figures: one showing a default layout without data in the output fields during the editing phase (figure 12; [180]), one showing a preview with data (figure 13; [181]) and the final version using the executable GUI code (figure 16; [181]). But one example is certainly not enough to illustrate all the symbols of figure 4, their underlying technical concepts and their visualisation.

3.9 The board considers not only the general view (i.e. the relationship between symbols and GUIs) to be insufficiently disclosed, but also the bridging steps of building an intermediate language (the XML-based canonical representation), of translating the symbols to that language and of translating the latter to executable GUI code.

3.10 As to the XML-based canonical representation, the term "*canonical*" generally designates some kind of standard form, i.e. a unique form for every object. Assuming that the term "canonic" is meant to be synonymous to "canonical" in the description, there is indeed a passage in the description ([70], last but one sentence) disclosing that the visual representation is

converted to a *single* canonic representation. But the next paragraph ([71], last sentence) reads:

> "In certain instances, a visual representation may be converted *to a plurality of canonic representations* prior to generating a plurality of user interface codes." (emphasis added)

3.11 This contradicts the interpretation of "canonical" to mean "standard form". But even if one assumes that the canonical representation is a standard form, there is no disclosure how to make the corresponding XML-based representation unique for any given visual representation.

3.12 Thus, the application does not define what is meant by "canonical" in this context, so that the skilled person implementing the invention would not know whether a particular representation he wants to choose is canonical or not.

3.13 As to the characterisation of the canonical representation as being "XML-based", it is well-known that the so-called *Extensible Markup Language (XML)* is essentially a framework in which a programmer can formulate his own XML language *instance*. He has to give a formal definition for his XML instance, for example a so-called XML schema. There are tools available which generate from an XML schema programs to validate the conformity of XML documents with the schema, or to parse them according to the schema.

3.14 However, the application fails to disclose such an XML schema. The only disclosure of the XML-based canonical representation are appendices A and B and figures 10B,

11B and 14, all of them relating to the visual representation in figure 11A (or its first development state in figure 10A). Notwithstanding the fact that large parts of these appendices are literally incomprehensible, absent further explanation, the board considers that one example of a visual representation and its corresponding XML-based canonical representation cannot provide a skilled person with sufficient information to enable him to create a whole XML language instance or its XML schema.

3.15 During oral proceedings, the appellant stated that a skilled person knows how to create a suitable XML instance and a corresponding XML schema.

3.16 The board agrees that a skilled person is able to create an XML schema *if* he has a specification of what should be modeled by the language. As stated above, the board does not see enough disclosure of the underlying technical concepts to create such a specification.

3.17 Furthermore, the XML-based canonical representation as well as the visual representation belongs to the core of the invention according to the appellant during oral proceedings. The board cannot see that a skilled person could develop such an important part of the invention by himself *without the exercise of inventive activity*, the latter being a prerequisite to sufficient disclosure.

3.18 As to the first translation from the visual representation to the XML-based canonical representation, there is not even one example of a predefined XML code portion to be included in the canonical representation when dragging and dropping symbols from the task panel

to the workspace. Nor is there any hint that the translation should be implemented in this way. The board has furthermore doubts that the creation of the canonical representation could function in such an easy way, since for example the GUI logic has to be expressed in some way in the canonical representation.

3.19 The appellant stated that paragraph [161], last but one sentence disclosed that the the canonical representation ("GML code") is automatically generated by the modeling system ("GM Storyboard"). Therefore the GUI logic is contained in the canonical representation.

3.20 The board notes that somehow the claimed method has to infer what the GUI logic is. A passage merely stating that this is done automatically does not help the skilled person to work out how this is done.

3.21 As to the appellant's argument that at least one XML schema for representing diagrams is well known, the board does not doubt this. However, merely representing a diagram does not mean that the XML schema can express the logic of the GUI to be generated, or that it is suitable to be translated into executable code.

3.22 As to this second translation from the XML-based canonical representation to executable GUI code, the appellant wrote in his letter dated 21 January 2014 (page 4, paragraph 3):

> "Similarly, translating an XML based representation into executable code would be immediately understandable for the skilled person from the present application. For example, such a translation may be carried out using Extensible

Stylesheet Language Transformations (XSLT). This is a known way of transforming XML documents into other objects, and can also be used to generate intermediate code (e.g. Java Byte code) or executable code."

3.23 During the oral proceedings, the board pointed out that XSLT is generally known to be a *turing-complete* programming language. So, one can program every computable method in XSLT, including a translation from the canonical representation to executable code. The question is how to do it, i.e. what is the program in XSLT which transforms a particular visual representation into executable code. The application doesn't even mention XSLT, let alone explain how it would be generated.

3.24 The appellant argued that figure 15 disclosed an example of GUI code in the Java programming language.

3.25 However, the board was not convinced that a very small portion of a Java program would give enough information to the skilled person to translate canonical representations in general into executable code.

3.26 The board notes that Rule 42(1)(e) EPC defines one task of a patent description as being to "describe in detail at least one way of carrying out the invention claimed, using examples where appropriate and referring to the drawings, if any". It follows that if the board points out that details are missing in the description, it is up to the appellant to show that the missing parts are common general knowledge. This the appellant has failed to do in the present case: There is not enough disclosure of the diagrammatic visual representation

language, of the XML-based canonical representation language and of the two translation steps yielding executable code.

3.27 Therefore, the invention as claimed in the second auxiliary request and thus (as argued above) in all requests is insufficiently disclosed in contravention of Article 83 EPC.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

T. Buschek D. Rees

Decision electronically authenticated