Internal distribution code:

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

Datasheet for the decision of 18 June 2019

Case Number: T 1651/15 - 3.5.07

Application Number: 09826753.7

Publication Number: 2356581

IPC: G06F17/30, G06F17/24

Language of the proceedings: EN

Title of invention:
Managing and automatically linking data objects

Applicant:
Ab Initio Technology LLC

Headword:
Data object linking/AB INITIO TECHNOLOGY

Relevant legal provisions:
EPC Art. 56

Keyword:
Inventive step - all requests (no)

Decisions cited:
T 1456/14, T 1255/16

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

Case Number: T 1651/15 - 3.5.07

D E C I S I O N
of Technical Board of Appeal 3.5.07
of 18 June 2019

Appellant: (Applicant)	Ab Initio Technology LLC 201 Spring Street Lexington, MA 02421 (US)
Representative:	Lloyd, Patrick Alexander Desmond Reddie & Grose LLP The White Chapel Building 10 Whitechapel High Street London E1 8QS (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 26 March 2015 refusing European patent application No. 09826753.7 pursuant to Article 97(2) EPC**

Composition of the Board:

Chairman	R. Moufang
Members:	R. de Man
	P. San-Bento Furtado

Summary of Facts and Submissions

I. The applicant (appellant) appealed against the decision of the Examining Division refusing European patent application No. 09826753.7, which was published as international application WO 2010/056867.

II. The decision cited, *inter alia*, the following document:

D5: "Protégé-OWL API Programmer's Guide", 21 September 2006, retrieved from http://web.archive.org/web/20080715124233/http://protege.stanford.edu/plugins/owl/api/guide.html.

The Examining Division decided that the subject-matter of claims 1 to 14 of the main request was not new in view of document D5 and that the subject-matter of claim 15 was not inventive. The subject-matter of claim 1 of each of the first to fifth auxiliary requests was also not new.

III. In its statement of grounds of appeal, the appellant maintained the requests considered in the decision under appeal as the main request and the second to sixth auxiliary requests and filed new first and seventh auxiliary requests. It submitted copies of the claims of all requests and the following document:

D8: "A Semantic Web Primer for Object-Oriented Software Developers", 9 March 2006 [retrievable from https://www.w3.org/2001/sw/BestPractices/SE/ODSD/20060117/].

IV. In a communication accompanying the summons to oral proceedings, the Board introduced the following documents:

D5a: "Protégé-OWL API Programmer's Guide", 21 September 2006, retrieved from https://web.archive.org/web/20070611201700/http://protege.stanford.edu/plugins/owl/api/guide.html;

D5b: "configuring-overview.png", 21 September 2006, retrieved from https://web.archive.org/web/20070611201727im_/http://protege.stanford.edu/plugins/owl/api/configuring-overview.png.

The Board expressed the preliminary view that the subject-matter of claim 1 of all requests lacked inventive step.

V. In a letter dated 17 May 2019, the appellant informed the Board that it would neither file written submissions nor attend the oral proceedings.

VI. Oral proceedings were held on 18 June 2019 in the appellant's absence. At the end of the oral proceedings, the chairman pronounced the Board's decision.

VII. The appellant requested that the decision under appeal be set aside and that a patent be granted on the basis of the main request or, in the alternative, one of the first to seventh auxiliary requests.

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

"A method for managing data objects stored in a data repository and displaying elements associated with data objects on a graphical user interface, the method including:

using reusable logic to provide a first displayed input element associated with a first data object stored in the data repository;

in response to user input associated with the first displayed input element, generating a second data object stored in the data repository;

linking the second data object with one or more related data objects in the data repository automatically based on the first data object; and

reusing the reusable logic to provide a second displayed input element associated with the second data object in which the second displayed input element is based on the one or more related data objects linked to the second data object."

IX. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that the "data repository" has been replaced with "scalable object-oriented database system".

X. Claim 1 of the second auxiliary request differs from claim 1 of the main request in that "information obtained from" has been inserted after "in which the second displayed input element is based on".

XI. Claim 1 of the third auxiliary request differs from claim 1 of the main request in that

"automatically navigating to the one or more related data objects linked to the second data object to gather information;"

has been inserted after "linking ... based on the first data object;" and in that

"based on the information gathered from the one or more related data objects linked to the second data object"

has been substituted for "based on the one or more related data objects linked to the second data object".

XII. Claim 1 of the fourth auxiliary request reads as follows:

"A method for managing data objects stored in a data repository and displaying elements associated with data objects on a graphical user interface, the method including:

using reusable logic to provide at least one displayed input element associated with a first data object stored in the data repository;

in response to user input associated with the displayed input element, generating a second data object and storing the second data object in the data repository;

linking the second data object with one or more related data objects in the data repository automatically based on the first data object;

displaying on the graphical user interface one or more elements associated with one or more data objects stored in the data repository and related to the second data object through one or more relational links;

wherein the one or more elements associated with the one or more data objects include an input element that provides a set of choices from which a user selects to provide information, in which the set of choices is based on the one or more data objects stored in the data repository and related to the second data object through the one or more relational links."

XIII. Claim 1 of the fifth auxiliary request differs from claim 1 of the fourth auxiliary request in that "storing the second data object" has been replaced with "storing the second dta [sic] object" and in that "information obtained from" has been inserted after "in which the set of choices is based on".

XIV. Claim 1 of the sixth auxiliary request differs from the fifth auxiliary request in that

"automatically navigating to the one or more related data objects linked to the second data object to gather information;"

has been inserted after "linking ... based on the first data object;" and in that "based on information obtained from" has been replaced with "based on the information gathered from".

XV. Claim 1 of the seventh auxiliary request differs from the main request in that "a script" has been inserted before "linking the second data object".

XVI. The appellant's arguments, where relevant to the decision, are discussed in detail below.

Reasons for the Decision

1. The appeal complies with the provisions referred to in Rule 101 EPC and is therefore admissible.

2. *The application*

2.1 The background section of the application explains that, in conventional data entry systems implemented as web applications, a user can generate new objects by clicking on a link. The link directs the user to a web page that lists the types of object that can be generated. When the user selects one of the listed types, he is taken to an editor that allows him to enter the necessary information about the new object, including links to existing objects. "Setting" these links to existing objects is said to often require either extensive navigation to find the link target or looking through a long list of existing objects.

2.2 The detailed description describes, with reference to Figure 1, a computing system 100 comprising a repository 104 that is referred to as an "Enterprise Meta Environment" (EME) and contains various types of metadata such as "documentation, record formats (e.g., fields and data types of records in a table), transform functions, graph jobs, monitoring information, etc." (page 4, line 27, to page 5, line 15, of the published application).

The computing system 100 also comprises a user-interface module 108, which generates an EME browser interface 202 as shown in Figure 2. To display details about a specific EME object, the interface 202 generates a browser page 206 as shown in Figure 3. This

page can contain various input elements for receiving user interaction (page 7, lines 10 to 23).

As described on page 9, lines 16 to 29, in some cases the system automatically links a newly created object to existing objects.

Main request

3. *Interpretation of claim 1*

3.1 Claim 1 of the main request is directed to a method for managing data objects in a data repository and displaying elements associated with data objects on a graphical user interface.

3.2 Claim 1 comprises steps of "using reusable logic" and "reusing the reusable logic" to provide first and second displayed input elements. The application as filed does not explain in detail what is meant by "reusable logic". The detailed description mentions the term only once on page 7, lines 24 to 26, where it refers to element renderer 204. The Board therefore interprets "reusable logic to provide a first/second displayed input element" as being an element renderer for rendering input elements of a graphical user interface.

3.3 Hence, the method of claim 1 first displays a first input element, such as a button, which is somehow associated with a first data object in the data repository.

3.4 In response to "user input associated with the first displayed input element", for example a button click, a

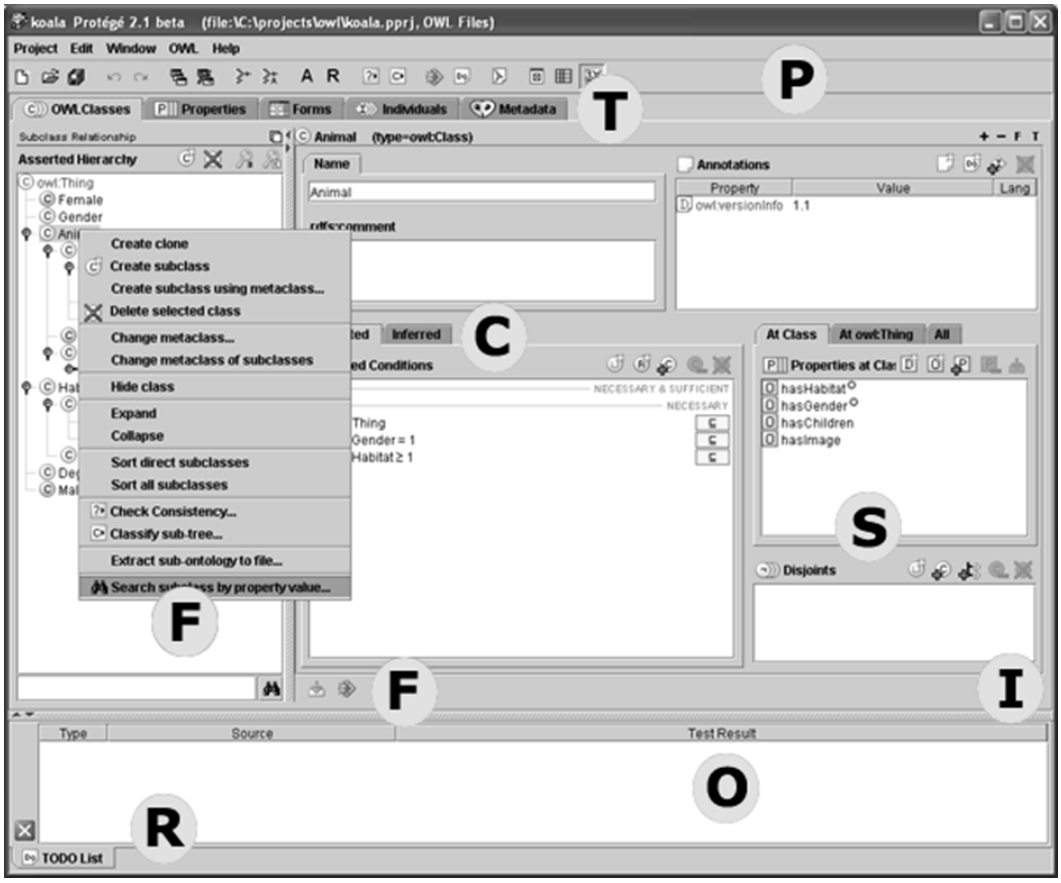
second data object is generated and stored in the data repository.

3.5 On the basis of the first data object, this second data object is automatically "linked" with one or more "related" data objects in the data repository. The claim includes no definition of the term "linking", which is therefore to be interpreted broadly.

3.6 Next, a second input element is displayed which is somehow associated with the second data object. This second input element is "based on the one or more related data objects linked to the second data object".

4. *Inventive step*

4.1 Document D5 is a printout of a web page archived by the Internet Archive. Because it is of relatively poor legibility, the Board introduced documents D5a and D5b into the proceedings. Document D5a is another printout of the same web page. Document D5b is a larger-sized copy of the screenshot shown on page 22 of document D5a. The screenshot is reproduced here:



4.2 Document D5/D5a relates to the "Protégé-OWL API", which is a Java library for the "Web Ontology Language" (OWL) (see D5a, page 1, section "Overview"). It contains a discussion of the Protégé-OWL editor, which provides editing and browsing facilities for "OWL models" (*ibid.*). The screenshot shown above depicts the user interface of this editor (D5a, page 22, section "Protégé Plug-in Development").

4.3 The Examining Division essentially argued that the screenshot, interpreted with the help of several passages from document D5, disclosed all the features of claim 1 in combination.

As is explained below, the Board does not fully agree with the Examining Division's analysis. But it does consider document D5/D5a to be a suitable starting

point for assessing inventive step of the claimed invention.

4.4 The screenshot shows a first input element (labelled "F") in the form of a context menu associated with a selected (first) data object that represents a "class" named "Animal". The user interface depicted in the screenshot shows and allows editing this "Animal class".

The skilled person would have understood that the context menu was rendered by an element renderer capable of rendering user-interface elements, i.e. by "using reusable logic" (see point 3.2 above).

The "Animal class" and other OWL classes editable by the Protégé-OWL editor correspond to "data objects" containing the definitions of these classes. These objects, together forming an "OWL model", are stored in an implicitly disclosed "data repository".

Document D5 thus discloses the step described in point 3.3 above.

4.5 The appellant argued that the "OWL classes" of document D5 could not be equated with the "data objects in an object-oriented database" of the claim. The correspondence between OWL classes and data objects would not have been immediately apparent to the skilled person, in particular in view of document D8, which included a table showing the differences between object-oriented languages and OWL. The appellant emphasised that, in an object-oriented database, a class is not itself a data object.

The Board agrees that document D5 is silent on the underlying repository being an object-oriented database. But so is claim 1 of the main request.

The Board does not agree that the OWL classes of document D5 are not "data objects" within the meaning of the claim. Document D5 discloses an editor for editing data objects that happen to represent OWL classes. These "OWL class" data objects have properties such as "Name" (see the screenshot). In the examples given in the application, data objects represent "departments", "employees", "notes" and "business rules". There is no reason why they cannot also represent OWL classes.

The appellant is correct that, in the context of object-oriented programming and object-oriented databases, a "class" of objects has to be distinguished from an "object" that is an instance of the class. But the Board does not equate OWL classes with objects that are instances of OWL classes. Instead, it merely recognises that an OWL class whose definition is being edited in the editor of document D5 is necessarily represented as a data structure which falls within the scope of the broad term "data object".

It having been established that the claim term "data object" objectively encompasses the OWL classes of document D5, it is unimportant whether a correspondence between the OWL classes of document D5 and the data objects of the claim would have been "immediately apparent to the skilled person" (cf. decisions T 1456/14 of 15 June 2018, reasons 3.3; and T 1255/16 of 15 February 2019, reasons 6.10).

4.6 The Examining Division reasoned that "in response to user input associated with the first displayed input element", i.e. in response to selecting the "Create subclass" entry in the context menu, a (second) data object representing a subclass of the "Animal" class would be created. This "subclass" data object would be displayed in the class hierarchy panel as a child of the "Animal class" data object. In other words, the second data object would be automatically linked with the first data object and thus "with one or more related data objects in the data repository ... based on the first data object".

The Examining Division further appears to have reasoned that selecting the "Create subclass" entry of the context menu would cause the user interface to switch to showing and allowing editing of the (second) subclass data object instead of the (first) "Animal class" data object. The "Properties at Class" widget, which in the screenshot is labelled "S" and which displays and allows editing a list of properties of the Animal class (as explained on page 22, last full paragraph, of document D5a), would then switch to displaying a list of properties of the (second) subclass data object and therefore be "a second displayed input element associated with the second data object". The Examining Division appears to have taken the view that those properties would have been inherited from the "Animal class", meaning that the widget would be "based on" the (first) "Animal class" data object to which the second data object is linked.

4.7 The Board agrees with the appellant that the Examining Division's interpretation of document D5 goes beyond its actual disclosure. In fact, since document D5 does not refer to user input activating the "Create

subclass" entry in the context menu, it does not disclose the feature "in response to user input associated with the first displayed input element, generating a second data object stored in the data repository". And as to what would happen if the user were to select the "Create subclass" entry, the Examining Division's novelty reasoning necessarily involves some amount of speculation.

4.8 Nevertheless, the Board considers that the skilled person, analysing the properties of the Protégé-OWL editor on the basis of the information contained in document D5/D5a, would have considerd it to be an obvious possibility that the editor's graphical user interface, in response to selecting the "Create subclass" entry, would behave as described in point 4.6 above.

4.9 Indeed, the skilled person would have expected that the newly created (second) subclass data object of the "Animal class" would be displayed in the class hierarchy panel as a child of the "Animal class" data object and thus automatically be linked with the first data object and hence "with one or more related data objects ... based on the first data object". This would have led the skilled person to the steps described in points 3.4 and 3.5 above.

4.10 The skilled person would also have expected that selection of the newly created subclass - whether that selection takes place automatically when the subclass is created or in response to user input - would switch the editor's user interface to showing that subclass and allow editing of it. This would mean that the "Properties at Class" widget, as the "second displayed input element", would switch to displaying a list of

properties of the subclass as "second data object". Since the skilled person would have been well aware that a subclass typically inherits the properties of its parent class, he would also have found it obvious that the properties listed in the "Properties at Class" widget would include the properties of the "Animal class" and thus be "based on" the first data object, which is "linked to the second data object". The skilled person would therefore also have been led to the step described in point 3.6 above.

4.11 In its statement of grounds of appeal, the appellant argued that the menu shown in the screenshot was not "based on one or more related data objects". However, the Board equates the "second input element" with the "Properties at Class" widget.

4.12 Hence, the subject-matter of claim 1 lacks inventive step over the disclosure of document D5/D5a (Article 56 EPC).

First auxiliary request

5. Claim 1 of the first auxiliary request adds that the data repository is a "scalable object-oriented database system". Since object-oriented database systems were well known at the priority date of the application, it would have been an obvious possibility to store the data objects of document D5 in such a database. The further feature specifying that the database is "scalable" represents an obvious desideratum.

The subject-matter of claim 1 of the first auxiliary request therefore lacks inventive step (Article 56 EPC).

Second and third auxiliary requests

6. Compared with claim 1 of the main request, claim 1 of the second auxiliary request clarifies that the second displayed input element is based "on information obtained from" the one or more related data objects linked to the second data object.

Claim 1 of the third auxiliary request adds that this information is obtained ("gathered") by "automatically navigating to the one or more related data objects linked to the second data object to gather information", i.e. the information "from the one or more related data objects" is obtained automatically.

These amendments do not affect the Board's interpretation of claim 1 and therefore do not overcome the inventive-step objection raised against claim 1 of the main request (Article 56 EPC).

Fourth auxiliary request

7. In addition to a number of minor reformulations which do not affect the claim's interpretation, in claim 1 of the fourth auxiliary request, the step "reusing the reusable logic ..." of claim 1 of the main request has been replaced with a step of displaying one or more "elements" associated with one or more data objects related to the second data object.

The elements include an "input element" that provides a set of choices "from which a user selects to provide information", the set of choices being based on the one or more data objects related to the second data object.

8. Following the Board's reasoning for the main request in point 4.10 above, the "Properties at Class" widget shown in the screenshot of document D5a is "associated with" the selected first data object (representing the "Animal class"). This first data object is a data object "related to the second data object through one or more relational links" (see point 4.9 above).

It is expected user-interface behaviour that selecting one of the property "choices" shown in the "Properties at Class" widget causes the user interface to "provide information" about the selected property.

9. The subject-matter of claim 1 therefore lacks inventive step (Article 56 EPC).

Fifth and sixth auxiliary requests

10. Compared with claim 1 of the fourth auxiliary request, claim 1 of the fifth auxiliary request clarifies that the "set of choices" is based "on information obtained from" the one or more related data objects linked to the second data object.

Claim 1 of the sixth auxiliary request adds that this information is obtained ("gathered") by "automatically navigating to the one or more related data objects linked to the second data object to gather information", i.e. the information "from the one or more related data objects" is obtained automatically.

These amendments do not affect the Board's interpretation of claim 1 and therefore do not overcome the inventive-step objection raised against claim 1 of the fourth auxiliary request (Article 56 EPC).

Seventh auxiliary request

11. Claim 1 of the seventh auxiliary request adds to claim 1 of the main request that the automatic linking is performed by a "script".

Since, at the priority date of the application, it was well known in the art to use scripts for performing programmatic actions, this amendment does not overcome the objection of lack of inventive step (Article 56 EPC).

Conclusion

12. Since none of the requests on file is allowable, the appeal is to be dismissed.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

I. Aperribay R. Moufang

Decision electronically authenticated