**Internal distribution code:**

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

# Datasheet for the decision of 03 March 2011

**Case Number:** T 0077/08 - 3.5.06

**Application Number:** 04001094.4

**Publication Number:** 1480121

**IPC:** G06F9/45, G06F17/60

**Language of the proceeding:** EN

**Title of invention:**
System and method for dynamic business logic integration

**Applicant:**
Millennium IT (USA) Inc.

**Headword:**
Business logic integration/MILLENNIUM

**Relevant legal provisions:**
EPC 1973 Art. 84, 56

**Keyword:**
Clarity (no) - all requests
Inventive step (no) - all requests

**Decisions cited:**

**Catchword:**

Europäisches Patentamt | European Patent Office | Office européen des brevets

Beschwerdekammern | Boards of Appeal | Chambres de recours

**Case Number: T 0077/08 - 3.5.06**

# D E C I S I O N
# of the Technical Board of Appeal 3.5.06
# of 03 March 2011

**Appellant:** (Applicant)
Millennium IT (USA) Inc.
245 First Street, 18th Floor
Cambridge, Massachusetts 02142 (US)

**Representative:**
Rupprecht, Kay
Meissner, Bolte & Partner GbR
Postfach 86 06 24
81633 München (DE)

**Decision under appeal:** **Decision of the Examining Division of the European Patent Office posted 02 July 2007 refusing European application No. 04001094.1 pursuant to Article 97(1) EPC.**

**Composition of the Board:**

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

## Summary of Facts and Submissions

I. This is an appeal against the decision of the examining division, announced in oral proceedings held on 12 June 2007, with written reasons dispatched on 2 July 2007, to refuse patent application number 04 001 094.4.
The claimed subject-matter of the various requests was found to be unclear in some cases. The subject-matter of the independent claims of all the requests was further found to lack novelty or an inventive step with respect to document:

D1 CA 2 351 990 A1, 26 December 2002.

II. Notice of appeal was filed in a letter dated 7 August 2007 and received on 8 August 2007. The fee was paid on 8 August 2007. A statement setting out the grounds of the appeal was filed on 12 November 2007 together with a main and three auxiliary requests, where the main request was identical to that of the impugned decision. A precautionary request for oral proceedings was made.

III. The board issued a summons to attend oral proceedings to be held on 3 March 2011. It gave its preliminary opinion that four terms used in the independent claims were not sufficiently disclosed by the description and not clear (Articles 83 and 84 EPC).
Furthermore, two amendments of claim 1 of all three auxiliary requests were objected to as violating the requirements of Article 123(2) EPC.
Finally, the board was of the preliminary opinion that the subject-matter of claim 1 of all requests appeared to lack an inventive step with respect to D1.

IV. In a letter dated and received on 3 February 2011, the appellant filed a new main request and three new

auxiliary requests. Moreover, it was requested to admit five new documents to the procedure. Only one of them is directly relevant to the present decision:

MBP1 Hudak, P.: "Conception, Evolution, and Application of Functional Programming Languages"; ACM Computing Surveys, Vol. 21, No. 3, September 1989; pages 359-411.

V. During oral proceedings held on 3 March 2011, the appellant filed a fourth, fifth and sixth auxiliary request.

VI. The appellant requests that the decision under appeal be set aside and that a patent be granted on the basis of the main request (claims 1-19), or alternatively on the basis of the first (claims 1-19), second (claims 1-18), or third auxiliary request (claims 1-18), all filed on 3 February 2011, as well as on the basis of the fourth (claims 1-19), fifth (claims 1-3; claims 4-19 to be adapted), or sixth auxiliary request (claims 1-18), filed during the oral proceedings.
The further text on file is: description pages 1-2, 4, 7-16 as originally filed; description pages 3, 3A-3C, 5, 6 as filed on 14 May 2007; and drawing sheets 1-26 as originally filed.

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

"1. A method for dynamically integrating a business logic rule into an application, the method comprising the steps of:
stating the business logic rule as an expression in a functional language that utilizes defined operators, functions and recognized or common terms in the

industry for which the application was written, and parameters that correspond to lookup fields in an associated database;
parsing the expression to produce an executable routine; and
providing the executable routine to the application."

Claim 1 of the *first auxiliary request* differs from claim 1 of the main request by adding to the first step of "stating" after the word "functions" the expression ", data types that are defined in the application" , and by adding at the end of the second step of "parsing the expression to produce an executable routine" the following sentence:

"that operates without re-compiling or re-writing the application".

Claim 1 of the *second auxiliary request* differs from claim 1 of the *main* request by adding to the first step of "stating" after the word "functions" the expression ", data types", and by adding at the end of the claim the following fourth and fifth steps:

"adding, via a dynamic scheme, new fields and sub-fields to the database; and
redefining one or more of said parameters by adding new database fields and sub-fields to associated tables that essentially link the parameters to the database."

Claim 1 of the *third auxiliary request* differs from claim 1 of the *second* auxiliary request by adding at the end of that claim the following sixth step:

"allowing a user to create new such keywords."

Claim 1 of the *fourth auxiliary request* differs from claim 1 of the *main* request by adding at the end of the second step of "parsing the expression to produce an executable routine" the following phrase:

"in the form of a parse tree when the business logic rule controls on-line actions".

Claim 1 of the *fifth auxiliary request* differs from claim 1 of the *main* request by adding at the end of that claim the following expression:

"said parameters being dynamically updated to correspond to changes in the underlying business logic rules".

Claim 1 of the *sixth auxiliary request* differs from claim 1 of the *third* auxiliary request by replacing at the end of that claim the word "keywords" by the expression "recognized or common terms in the industry for which the application was written", and by adding at the end of that expression the following phrase:

"wherein the database is dynamically configurable".

In all requests, claim 12 is the corresponding independent system claim.

VIII. At the end of the oral proceedings the chairman announced the board's decision.

## Reasons for the Decision

1. **Admissibility of the appeal**

The appeal satisfies the requirements of the EPC for admissibility, see sections I and II above.

2. **Original disclosure and admissibility of the requests**

2.1 The objections with respect to Article 123(2) EPC raised by the board in its summons to oral proceedings, sections 5.5 and 5.6, respectively, were remedied in the final requests by replacing and by re-introducing, respectively, the expressions in question.

2.2 Claim 1 of the *main request* differs from the preceding main request by replacing the term "keywords" by the expression "recognized or common terms in the industry for which the application was written". The latter expression is based on the original description, page 8, lines 7-10.

2.3 This amendment had also been applied to claim 1 of the *first to third auxiliary request*, with the exception of the second occurrence of the term "keywords" in the third auxiliary request, independent claims 1 and 12, last line. The appellant stated during oral proceedings that this happened by mistake and that they were willing to remedy it if that request was otherwise allowable.

2.4 As to claim 1 of the *fifth auxiliary request*, the appellant indicated the passage in the original description, page 4, line 8-15 as the basis for adding to the claim the expression "said parameters being dynamically updated to correspond to changes in the

underlying business logic rules".

However, the expression "business logic rules" is used in the plural form in both the passage and the added expression, whereas the rest of the claim uses it in the singular. Secondly, the claim (including the added expression) deals with "parameters" whereas the passage deals with a "set of parameters". Thirdly, different parameters are meant in the two contexts. The first two sentences of the passage read:

> "The set of parameters associated with a given GUI is dynamically updated, to correspond to changes in the underlying business logic rules. The user writes and/or edits the expressions for the respective business logic rules by selectively combining the available functions, operators and parameters."

So, the set of parameters that a graphical user interface (GUI) *offers* for writing or editing business logic rule expressions is updated. In the claim, one business logic rule expression which is already written "*utilizes*" - among other things - specific parameters. These are then updated, according to the claim amendment. However, the choice of parameters offered by the GUI had already been made for the specific business logic rule expression of the claim. So, the claim does not specify the same point in the process for updating as does the cited passage.

Therefore, the amendments of claim 1 of the fifth auxiliary request violate the requirements of Article 123(2) EPC, and consequently, this request is *not admitted* to the procedure.

2.5 As to the *other requests*, taking into account the criteria given in the Rules of Procedure of the Boards of Appeal, Article 13(1), the board exercised its discretion to admit them.

3. **Clarity of claim 1 of all requests**

The board objected in its summons to oral proceedings, section 4 that the following four terms, used in the claims, were ambiguous and insufficiently disclosed, which resulted in a lack of clarity of the claims (Article 84 EPC), namely:

- business logic rule;
- keywords;
- parsing the expression (in a functional language) to produce an executable routine;
- expression in a functional language.

3.1 As to "*business logic rule*", the board has come to the conclusion that a skilled person would recognise that the technical content of this term is simply the same as that of the term "rule", which is considered to be clear.

3.2 The term "*keywords*" had been replaced in all the present requests by "*recognized or common terms in the industry for which the application was written*" (with the exception of one occurrence by mistake in the third auxiliary request - see section 2.3 above). The board considers this expression adequately clear and realisable by the skilled person, at least in the form of *variable names*. For example, if a program is written for the image processing industry, the programmer might well use the variable name "GammaCorrection", a common term in the industry. The board notes that it considers

that such a use would satisfy this feature of the present claims.

3.3 As to the term "*parsing the expression to produce an executable routine*", the board has come to the conclusion, partly based on the documents submitted by the appellant with its response to the summons, that it agrees with appellant that this feature would be understood, and is realisable, by the skilled person.

3.4 However, the board maintains its clarity objection to the phrase "*expression in a functional language*", which is still used in the independent claims of all the present requests. The board accepts that "functional language" is a *well-known* term (having been used since the 1950's). It is, however, *not well-defined*. It is not clear which programming languages fall under this term and which do not. There are functional elements in almost any language, even in imperative ones. It is possible to program in a functional style even in an imperative language like "C". This means that the word "functional" is more a matter of style and programming paradigm than an actual definition of class of languages. See for example the document MBP1 (introduced by the appellant), page 361, right column, last but one paragraph:

> "Since most languages have expressions, it is tempting to take our definitions literally and describe functional languages via derivation from conventional programming languages: Simply drop the assignment statement and any other side-effecting primitives. This approach, of course, is very misleading. The result of such a derivation is usually far less than satisfactory, since the purely functional subset of most imperative languages is hopelessly weak (although

there are important exceptions, such as Scheme [Rees and Clinger 1986]).
Rather than saying what functional languages don't have, it is better to characterize them by the features they do have. For modern functional languages, those features include higher-order functions, lazy evaluation, pattern matching, and various kinds of data abstraction - all of these features will be described in detail in this paper."

And it continues on page 362, paragraph 2:

"This discussion suggests that what is important is the functional programming *style*, in which the above features are manifest and in which side effects are strongly discouraged but not necessarily eliminated."

So, there are gradations as to how "functional" a language is, from purely functional to "less functional", with or without features like multiple assignments to variables, higher-order functions, lazy evaluation, or pattern matching.

If one takes the language "LISP", for example, which is often called functional, but with which you can also program in a "non-functional style", it is not clear if LISP would be covered by the term "functional language" in the claim, or not.

Therefore, the subject-matter of claim 1 of all requests is not clear (Article 84 EPC).

4. This conclusion suffices to dismiss the appeal.

4.1 However, since the question of inventive step had been extensively discussed in the first instance, the board

will give its assessment of that question. Given the above considerations with respect to "functional language" and the fact that the application discloses only fragmentary, incomplete, examples of an "expression in a functional language" (e.g. figure 3, element 36 and figure 4A, element 36), the board assumes that the skilled person might consider or interpret that this feature is to be taken as an "*expression in a programming language*".

## 5. **Inventiveness of claim 1**

### 5.1 **Main request**

5.1.1 The appellant alleged that the main difference between claim 1 and the disclosure of D1 is that in the claim a business logic rule is stated as an "*expression in a functional language*", whereas in D1 it is formulated as a *script* in a file in the *markup language XML* which is then automatically included in a *source code template* (see D1, figures 4A and 4B) which is preferably written in Java (see D1, page 12, lines 15-17).

5.1.2 He further stated that the method of D1 had a low flexibility which led to the objective technical *problem* of the invention with respect to D1 of how to *provide more flexibility* when writing business logic rules.

For a number of reasons, the board is unconvinced by this reasoning:

5.1.3 As stated above in section 4., the board considers the "expression in a functional language" in claim 1 simply as an "expression in a programming language".

The skilled person would understand the term "expression" in a programming context as an entity which evaluates to a result (with or without side-effects on the state of the computer).

5.1.4 Secondly, the board is of the opinion that any activity of writing text in a form which is *automatically executable* by a computer is a kind of *programming*, regardless whether this is done in a source code file or in an XML file.

5.1.5 Thirdly, there is one embodiment in D1, page 10, line 26 and on figure 3A where the user writes the business logic rules *directly in source code* without writing an XML script:

"To create new rules for placement in group 388, a user writes source code 391 for a rule and then uses compiler 392 to compile code 391 to created executable code 393 which is then subsequently placed in group 388."

5.1.6 The business logic rule that is disclosed in D1, figure 4A, element 410 (as a script in an XML file) reads as follows:

```
"<SCRIPT>
    if (!k_eq.equals(fiType)) {
        %MSG_REDIRECT
    }
 </SCRIPT>"
```

This is an imperative if-then-else statement and so does not qualify as an "expression in a programming language".

5.1.7 Thus, the board recognises as the *difference* between claim 1 with D1 that claim 1 expect the user to formulate a business logic rule as an *expression* in a programming language" whereas the method of D1 expects a *statement* in a programming language for that purpose.

5.1.8 As to the alleged technical problem of how to provide more *flexibility*, it is questionable to what extent making a system more convenient or "flexible" is a technical issue. Secondly, regardless whether it is a technical one or not, the board does not consider the usage of an expression to be generally more flexible than a statement when formulating a business logic rule.

5.1.9 Furthermore, since almost all programming languages provide expressions, a skilled person would implement the ability for the user to formulate a business logic rule as an expression if the skilled person considers this formalism more adapted for the expected kind of business logic rules, without exercising inventive skills.

5.1.10 The appellant further argued that claim 1 and D1 differed in that the claimed method did not need *linking* of compiled program parts so that new business logic rules can be *added without stopping* the application.

5.1.11 The appellant indicated three passages in the original description, page 3 in order to show the adding of business logic rules without stopping the application. They read as follows:

- "dynamically integrating changes in the rules" (line 15);

- "the application software readily incorporates the new and/or revised rules" (line 20); and
- "The DBLRI translates the new expressions into corresponding executable routines that are then available to the application software." (line 25).

However, these passages do not actually disclose an adding of business logic rules without stopping the application. Neither can the board identify any passage in the application which would clearly and unambiguously disclose, or even imply, this feature. Moreover, even if such a feature were disclosed in the application as a whole, the subject-matter of the independent claim would not imply it.

5.1.12 As to not needing linking, what is actually claimed is "*dynamically integrating*" (line 1). However, a dynamic integration of a portion of a program into a whole program was well-known at the relevant priority date of the application, as is evidenced by the application's reference to "DLLs" (Dynamic Link Libraries) without further explanation. There is nothing in D1 which would deter the skilled person from using this well-known technique.

5.1.13 Therefore, the subject-matter of the claim 1 of the main request is not inventive over the disclosure of D1 (Article 56 EPC).

5.2 **First auxiliary request**

As noted above in section VII, claim 1 of the first auxiliary request differs from claim 1 of the main request by adding to the first step of "stating" after the word "functions" the expression ", *data types that are defined in the application*" , and by adding at the

end of the second step of "parsing the expression to produce an executable routine" the following sentence:

"*that operates without re-compiling or re-writing the application*".

The appellant did not dispute that it is commonplace for programming languages to include the ability to define data types. The board therefore considers it obvious to use data types defined in the application in order to write business logic rules for that application.

As to the second additional feature, document D1, page 9, lines 8-11 discloses that the source code of the application is compiled once, and that it is not required to be modified if new business logic rules are added.

Therefore, the subject-matter of that claim is not inventive over the disclosure of D1 (Article 56 EPC).

5.3 **Second auxiliary request**

The appellant stated that the main difference of claim 1 of the second auxiliary request to D1 was the possibility that the functional language was extensible via the database. However, the board considers the *addition of new fields to a database* a commonplace feature, especially in the field of business related programs.

Therefore, the subject-matter of that claim is not inventive over the disclosure of D1 (Article 56 EPC).

5.4 **Third auxiliary request**

The appellant stated that the main difference of claim 1 of the third auxiliary request to D1 was the possibility that the functional language was extensible by *creating new "keywords"*. However, what is actually claimed is "in a ... language that utilizes ... recognized or common terms in industry for which the application was written". As noted above at section 3.2, this feature is actually satisfied by the use of appropriate variable names, a commonplace feature in any programming language. Moreover, the use of variable names cannot be said to "extend" a language.

Therefore, the subject-matter of that claim is not inventive over the disclosure of D1 (Article 56 EPC).

5.5 **Fourth auxiliary request**

As noted above in section 3.3, the board agrees with the appellant that the intermediate step of interpreting a parse tree directly was well-known at the priority date of the application.

It further considers it obvious for a skilled person to directly interpret urgent business logic rules, as for example for on-line actions on the stock market, instead of compiling them, and arbitrarily to choose the well-known parse tree as the intermediate code format for that purpose.

Therefore, the subject-matter of that claim is not inventive over the disclosure of D1 (Article 56 EPC).

5.6 **Sixth auxiliary request**

As mentioned before, claim 1 of the sixth auxiliary request differs from claim 1 of the third auxiliary request in that the term "keyword" had been entirely replaced and by the additional feature of a *dynamic configurability of the database*. The appellant indicated the passage on page 3, line 30 as the basis in the original description for that. However, this passage merely discloses that new fields can be added to the database. This feature is already present in the third auxiliary request.

In order to show the disclosure *how* the database is configured and under what circumstances, the appellant further indicated the passage on page 6, line 28 which states that a *dynamic scheme* is used for adding new fields. This feature is again already present in the third auxiliary request, so that the same reasoning as for the third auxiliary request applies to the sixth auxiliary request.

Therefore, the subject-matter of that claim is not inventive over the disclosure of D1 (Article 56 EPC).

**6. Conclusion**

Thus, the subject-matter of the independent claims of all the requests is unclear, in violation of Article 84 EPC. Hence, no request is allowable and the appeal must be dismissed.
Further, as shown in the above discussion of a possible interpretation of the claimed subject-matter, no request satisfies the requirements for an inventive step, in violation of Articles 52(1) and 56 EPC.

## Order

**For these reasons it is decided that:**

The appeal is dismissed.

The Registrar: The Chairman:

B. Atienza Vivancos D. Rees