Internal distribution code:

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

Datasheet for the decision of 23 November 2012

Case Number:	T 1893/08 - 3.5.06
Application Number:	01115100.8
Publication Number:	1178404
IPC:	G06F 9/45
Language of the proceedings:	EN

Title of invention:
Method and system for compiling multiple languages

Applicant:
MICROSOFT CORPORATION

Headword:
Compiling multiple languages/MICROSOFT

Relevant legal provisions (EPC 1973):
EPC Art. 84, 56, 111(1)

Keyword:
"Claims - clarity (yes) after amendment"
"Inventive step - (yes) after amendment"
"Decision re appeals - remittal (yes)"

Decisions cited:
-

Catchword:
-

This datasheet is not part of the Decision.
It can be changed at any time and without notice.

Case Number: T 1893/08 - 3.5.06

DECISION
of the Technical Board of Appeal 3.5.06 of 23 November 2012

Appellant: (Applicant)	MICROSOFT CORPORATION One Microsoft Way Redmond, Washington 98052-6399 (US)
Representative:	Grünecker, Kinkeldey Stockmair & Schwanhäusser Leopoldstraße 4 D-80802 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted 10 March 2008 refusing European patent application No. 01115100.8 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman:	W. Sekretaruk
Members:	A. Teale M. Müller

Summary of Facts and Submissions

I. The appeal is against the decision by the examining division, dispatched on 10 March 2008, to refuse European patent application No. 01 115 100.8 *inter alia* on the basis that the subject-matter of claim 1 according to the then first, second and third auxiliary requests did not involve an inventive step, Article 56 EPC 1973, in view of the following document:

D1: US 6 067 413 A.

The following document was also mentioned in examination proceedings:

D2: GB 1 367 741 A.

II. In a notice of appeal, received on 20 May 2008, the appellant requested that the decision be set aside and that a patent be granted. The appellant also made an auxiliary request for oral proceedings. The appeal fee was paid on the same day.

III. With a statement of grounds of appeal, received on 18 July 2008, the appellant filed amended claims according to a main request and auxiliary requests I to III. The appellant requested that the decision be set aside and that a patent be granted on the basis of the claims according to said main request and said auxiliary requests I to III. If the board was unable to set aside the decision or grant the main request in written proceedings then oral proceedings were requested.

IV. In an annex to a summons to oral proceedings the board set out its preliminary opinion on the appeal, expressing doubts *inter alia* as to the clarity of the claims, Article 84 EPC 1973, and as to whether the claimed subject-matter involved an inventive step, Article 56 EPC 1973.

V. With a letter received on 19 October 2012 the appellant filed amended claims according to a main and an auxiliary request and requested that the decision be set aside and a patent granted on the basis of the claims according to said main or said auxiliary request.

VI. In the oral proceedings held on 23 November 2012 the board expressed doubts as to the clarity of the claims and as to whether the claimed subject-matter involved an inventive step. The appellant then submitted amended claims and description pages according to a new sole main request. The appellant requested that the decision under appeal be set aside and that a patent be granted in the following version:

Description:
Pages 1, 3, 4, 6, 8, 9, 11 to 16 and 18, as originally filed.
Pages 2, 2a and 10, received on 23 October 2006.
Page 17, received on 9 November 2007.
Pages 2b, 5 and 7, received on 23 November 2012.

Claims:
1 to 9, received on 23 November 2012.

Figures:
2 and 6, as originally filed.
1, 3 to 5 and 7, received on 23 October 2006.

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

VIII. The independent claims according to the main request read as follows:

"1. A method of producing a common language file and compiling a computer program written in a first source language and having an import statement that imports the common language file, said method comprising: creating, by a compiler for a second source language different from the first language, the common language file using a source language file in the second language; parsing the computer program; examining each statement during the parsing act and determining if the statement is an import statement related to the common language file; if the statement relates to the common language file, reading the common language file into a symbol table; if the statement relates to a first language symbol table entry, adding the information into the symbol table; and if the statement relates to output generation, supplying the statement to a back end portion of a compiler for performing output synthesis, wherein the common language file is represented in a different language than the first and second language; wherein reading the common language file comprises: parsing the common language file, and adding type and method information in metadata in the common language file to the symbol table." and

"8. A system (34) comprising a first and a second compiler front end system for generating code to be used by an execution environment (32), both compiler front end systems comprising: a metadata module (33) that is configured to compile information to produce metadata information (28); and a code module (35) that is configured to compile information to produce executable instructions (30); wherein both compiler front end systems are configured to produce metadata information (28) and executable instructions (30) as the result of compiling a source file in a first and a second language, respectively; wherein the second language is a source language which is different from the first language, wherein the first compiler front end system consumes metadata information of a common language file produced by the second compiler front end system as a result of compiling the source file in the second language, wherein consuming metadata information produced by the second compiler front end system comprises parsing the metadata information, and converting type and method information in the metadata information into a form for a symbol table corresponding to the source file in the first language; wherein the common language file is represented in a different language than the first and second language."

Reasons for the Decision

1. *Admissibility of the appeal*

In view of the facts set out at points I to III above, the appeal is admissible.

2. *The context of the invention*

2.1 The invention as claimed (see figure 2) relates to compiling a computer program written in two source languages, for instance Lisp and C++, into object code which can be executed by computing hardware. The use of two source languages allows a programmer to select the source language most suited to expressing the necessary concepts and functionality. In general, the compiling process involves two parts: a "front end" portion for analyzing the source code to yield what is termed a "common language file", and a "back end" portion for synthesizing object code from the common language file.

2.2 The invention involves using a different compiler front end system for each of the two source languages. The language of the common language file differs from the two source code languages. The first compiler front end system is adapted to import a compiled common language file produced by the second compiler front end system.

2.3 As shown in figure 2, a common language file comprises metadata and executable instructions, produced by a metadata module and a code module, respectively, of a compiler front end system, the metadata describing the executable instructions in terms of the types, classes, methods, properties and events in the executable instructions. According to the application, the executable instructions can be Java bytecode or p-code.

2.4 The compilation of a program written in a first and a second source language commences by the second compiler front end system (36) compiling the source code written in the second source code language (38) into a common language file (44). As shown in figure 2, this file is then used in the subsequent compilation of the source code written in the first source code language (48). As shown in figure 6, during the subsequent compilation the source code is parsed and analysed. If an "import statement" is encountered, then the common language file is read and parsed by the first compiler front end system with respect to the grammar of the common language. The effect of the import statement is that type and method information in metadata in the common language file is added to the symbol table produced by the first compiler front end system, a symbol table comprising all the entities such as variables, objects and data structures referred to during the compilation process. If the statement relates to a first language symbol table entry, then the information is added into the symbol table. Other types of statements result in the synthesis of object code by the back end portion of a compiler.

3. *The admittance of the main request submitted in the oral proceedings before the board*

3.1 The question of the admittance of this request, involving amendments to the claims and description, turns on whether the board allows the corresponding amendment to the appellant's case, Article 13 RPBA (Rules of Procedure of the Boards of Appeal, OJ EPO 2007, 536). According to Article 13(1) RPBA, any amendment to a party's case after it has filed its grounds of appeal or reply may be admitted and considered at the board's discretion. The discretion shall be exercised in view of *inter alia* the complexity of the new subject-matter submitted, the current state of the proceedings and the need for procedural economy. Under Article 13(3) RPBA, amendments sought to be made after oral proceedings have been arranged shall not be admitted if they raise issues which the board cannot reasonably be expected to deal with without adjournment of the oral proceedings.

3.2 In the present case the board is satisfied that the amendments made to the claims and description were in reaction to objections raised by the board in the oral proceedings. The limited extent of the amendments and the explanations provided by the appellant were such that the board was readily able to assess the effect of the amendments without adjournment of the oral proceedings.

3.3 Hence the board admitted the main request into the proceedings.

4. *The amendments to the application*

4.1 *The claims*

Independent claims 1 and 8 are based on claims 17 and 1 as originally filed, respectively, in each case restricted using features taken from figure 2 and its description on page 7, line 16, to page 8, line 19, as originally filed.

4.2 *The description and figures*

Editorial amendments aside, page 2 has been amended and 2a added to acknowledge prior art documents, Rule 27(1)(b) EPC 1973. Page 2b has been added and pages 5 and 7 amended to adapt the description to the claims, Rule 27(1)(c) EPC 1973. The amendments to figures 1 and 3 are based on page 7, lines 2 to 4, and page 4, lines 8 to 14, as originally filed, respectively.

4.3 *The requirements of Articles 123(2) EPC and 84 EPC 1973*

The board is consequently satisfied that the amendments to the application comply with Article 123(2) EPC as regards added subject-matter and that the claims are now clear, Article 84 EPC 1973.

5. *Document D1*

5.1 D1 forms the closest prior art on file and relates to a computer program written in the two object-orientated source languages C++ and Java; see figure 3, items 200 and 100, respectively. To achieve the necessary interoperability between programming languages, D1 teaches to exploit the "congruency" between the more general language C++ and the more specialized language Java; see column 2, lines 7 to 18, and figure 1. This allows a common representation of data in what is termed a "persistent symbol table" to be shared between the compilers during program compilation and also to act as a common runtime representation of data between the different languages when the program is run; see column 2, lines 25 to 46, and claims 1 and 2 of D1. Figure 3 shows the case of a class declared in C++ ("class B") being derived from a class ("class A") declared in Java; see column 4, lines 22 to 30. The board understands the two compilers in D1 to share data via the "persistent symbol table" in computer memory (typically in RAM), this data structure also being accessed by the resulting program when it is run.

5.2 It is implicit in D1 that each of the Java and C++ compilers comprises a compiler front end portion. It is further implicit in the compilation of the C++ source language program that the program is parsed and that certain program statements, such as variable declarations, cause information to be added into the C++ compiler symbol table. Other statements, in particular program instructions, relate to output generation and are supplied to a back end portion of the compiler to generate code for use by an execution environment, thus resulting in the first and second compiled parts mentioned in claims 1 and 7 of D1. The persistent shared symbol table known from D1 comprises information on data definitions and thus metadata.

5.3 Hence, in terms of claim 1 of the main request, D1 discloses a method of producing information on data definitions and compiling a computer program written in a first source language (C++), said method comprising: creating, by a compiler (101) for a second source language (Java) different from the first language, said information on data definitions using a source language file in the second language; parsing the computer program (200); examining each statement during the parsing act and, if the statement relates to a first language symbol table entry, adding the information into the symbol table and, if the statement relates to output generation, supplying the statement to a back end portion of a compiler for performing output synthesis.

5.4 In terms of claim 8 of the main request, D1 discloses a system (see figure 3) comprising a first (201) and a second (101) compiler front end system for generating code to be used by an execution environment, wherein both compiler front end systems are configured to produce metadata information and executable instructions as the result of compiling a source file in a first (C++) and a second language (Java), respectively (see, in particular, page 7, lines 21 to 23, of the description); wherein the second language is a source language which is different from the first language, wherein consuming metadata information produced by the second compiler front end system comprises converting type and method information in the metadata information into a form for a symbol table corresponding to the source file in the first language.

6. *Inventive step, Article 56 EPC 1973*

6.1 *Claim 1*

6.1.1 In the light of the above analysis, the subject-matter of claim 1 differs from the disclosure of D1 in the following features:

a. the information on data definitions is in the form of a common language file represented in a different language to the first and second languages;

b. the first source language has an import statement that imports the common language file and

c. as part of said examination, determining if the statement is an import statement related to the common language file and, if so, reading the common language file into a symbol table by parsing the common language file and adding type and method information in metadata in the common language file to the symbol table.

6.2 *Claim 8*

The subject-matter of claim 8 differs from the disclosure of D1 in the following features:

a. the first compiler front end system consumes metadata information of a common language file produced by the second compiler front end system as a result of compiling the source file in the second language, the common language file being represented in a different language to the first and second languages;

b. said consumption comprises parsing the metadata information and

c. both compiler front end systems comprise a metadata module that is configured to compile information to produce metadata information and a code module that is configured to compile information to produce executable instructions.

6.3 Difference features "a" of claims 1 and 8 set out corresponding method and apparatus features, respectively. According to D1, information on data definitions is made available by the Java compiler to the C++ compiler (considered to be the second and first compiler front end systems, respectively, in terms of claims 1 and 8) by means of a "persistent symbol table" in computer memory which also acts as a common runtime representation; see column 2, lines 25 to 55. In contrast, difference feature "a" according to both claims 1 and 8 sets out the information on data definitions being in the form of a common language file which is produced by the second compiler and consumed by the first compiler. Hence the sharing of data between the compilers in D1 using computer memory is replaced by the first compiler reading in a file produced by the second compiler directly. In the board's view, the term "file", in the context of the application, implies data stored sequentially as a copyable product in a file system, typically in peripheral storage (such as a hard disk), rather than in a non-sequential graph structure (see D1, figure 3 and column 3, line 62, to column 4, line 21) in computer memory (such as RAM). From the perspective of D1, the claimed common language file allows the information on data definitions from one compiler to be stored and transferred as a copyable product for use by another (or the same) compiler later and on another (or the same) computer. Thus the compilation of a program written in two source languages can be interrupted after the first compilation has taken place, meaning that the result of the first compilation can, for example, be distributed with the second compiler so that the first and second compilations need not be temporally contiguous or occur on the same computer.

6.4 The appellant argued in the oral proceedings, and the board agrees, that the objective technical problem solved by difference feature "a" of claims 1 and 8 over D1 is "how to compile a program using data definitions from another compiler with greater flexibility". The use of the common language file represented in a different language to the first and second languages, as set out in difference feature "a" of claims 1 and 8, solves this problem in a way which does not require the first and second compilations to be temporally contiguous or that they occur on the same computer, thus increasing the flexibility of compilation. This solution of the objective technical problem is neither known from, nor specifically hinted at, by any document on file. Moreover difference feature "a" cannot be simply added to the system known from D1; its incorporation would mean that *a*, if not *the*, crucial feature of the system of D1 would have to be dispensed with, namely the common runtime representation of data shared between the compilers and the runtime environment. The board therefore concludes that it would not have been obvious for the skilled person to modify the system of D1 so as to incorporate difference feature "a".

6.5 For the sake of completeness, the board points out that the subject-matter of claims 1 and 8 involves an inventive step, Article 56 EPC 1973, in view of two further prior art disclosures discussed in these appeal proceedings, which, due to the amendments made to the claims during the course of these appeal proceedings, ultimately proved to be less relevant to inventive step than the main disclosure in D1 discussed above.

6.6 In the oral proceedings the relevance to inventive step of the prior art acknowledged in D1 in column 1, lines 48 to 52, was discussed. This passage mentions a program written in Java being expressed in C, the board understanding this to mean that a Java program is compiled into C and then subsequently compiled into object code. The board thus takes the view that at the priority date it would have been straightforward to combine source code written in C with source code written in Java by using such a compilation. In this scenario C is considered as both the claimed first source language and the language of the common language file, whilst Java forms the second source language. The subsequent C compiler would have acted on the C "#include" directive by reading in data resulting from the compilation of the Java source code. In order to have arrived at the subject-matter set out in claims 1 and 8, the skilled person would have had to choose a language for the common language file different from the first and second source languages. The board is of the opinion that the skilled person, on the basis of the brief section in D1, would have found the use of C as the "intermediate language" sufficient to achieve interoperability between the programming languages C and Java and would thus have had no reason to introduce another, different common language to replace C in this situation. Consequently the subject-matter set out in claims 1 and 8 also involves an inventive step, Article 56 EPC 1973, in view of this disclosure in D1.

6.7 The claimed subject-matter also involves an inventive step, Article 56 EPC 1973, in view of the disclosure of D2. D2 aims to avoid having to write a complete new compiler for each new high level source language; see page 2, left column, lines 60 to 64. This is achieved by constructing a compiler as a source-languagedependent translation step into an intermediate language (see page 1, lines 14 to 25) followed by a fixed "problem transformation" step into machine code; see page 1, lines 26 to 36, and page 2, lines 78 to 93. While the translation step and problem translation steps known from D2 can be considered as being carried out by a compiler front end system and compiler back end system, respectively, the board notes that every complete compiler according to D2 operates on a single language only and that D2 does not mention the problem of interoperability between programming languages. Even if the skilled person had addressed the problem of interoperability between two languages, the board considers that he/she would not have arrived at the claimed solution in an obvious manner, namely to adapt one compiler to read in an intermediate language file in response to an import statement, *inter alia* in view of alternative solutions such as those disclosed in D1 and discussed above. The prior art documents on file would not have provided the skilled person with a hint to modify the disclosure of D2 to use two source languages and thus two compiler front end systems or to adapt one compiler front end system to read-in a file produced by another compiler front end system in response to an import statement.

6.8 Hence the board finds that the subject-matter set out in claims 1 and 8 involves an inventive step, Article 56 EPC 1973.

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.

2. The case is remitted to the examining division with the order to grant a European patent with the following documents:

Description:
Pages 1, 3, 4, 6, 8, 9, 11 to 16 and 18, as originally filed.
Pages 2, 2a and 10, received on 23 October 2006.
Page 17, received on 9 November 2007.
Pages 2b, 5 and 7, received on 23 November 2012.

Claims:
1 to 9, received on 23 November 2012.

Figures:
2 and 6, as originally filed.
1, 3 to 5 and 7, received on 23 October 2006.

The Registrar: The Chairman:

G. Nachtigall W. Sekretaruk