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

Datasheet for the decision of 9 November 2012

Case Number: T 0505/09 - 3.5.06

Application Number: 04810843.5

Publication Number: 1683025

IPC: G06F 11/36

Language of the proceedings: EN

Title of invention:
System, method, and computer program product for identifying code development errors

Applicant:
Siemens Product Lifecycle Management Software Inc.

Headword:
Testing program code/SIEMENS

Relevant legal provisions (EPC 1973):
EPC Art. 56

Keyword:
"Original disclosure and clarity - after amendment (yes)"
"Inventive step over general common knowledge (yes)"
"Common knowledge in the pertinent technical field insufficiently substantiated"
"Remittal for further prosecution"

Europäisches Patentamt | European Patent Office | Office européen des brevets

Beschwerdekammern | Boards of Appeal | Chambres de recours

Case Number: T 0505/09 - 3.5.06

D E C I S I O N
of the Technical Board of Appeal 3.5.06
of 9 November 2012

Appellant: (Applicant)
Siemens Product Lifecycle Management Software Inc.
5800 Granite Parkway, Suite 600
Plano, TX 75024 (US)

Representative:
Kley, Hansjörg
Siemens AG
Postfach 22 16 34
D-80506 München (DE)

Decision under appeal:
Decision of the Examining Division of the European Patent Office posted 7 August 2008 refusing European patent application No. 04810843.5 pursuant to Article 97(2) EPC.

Composition of the Board:

Chairman: D. H. Rees
Members: M. Müller
C. Heath

Summary of Facts and Submissions

I. The appeal lies against the decision of the examining division, with written reasons dispatched on 7 August 2008, to refuse European patent application 04810843.5. The decision mentions, *inter alia*, the following two documents based on which objections had been raised during examination:

D1: Greer D., "How to debug a program", excerpt from the "SMUG" book, pp. 1-2, March 2001, retrieved from the Internet, and

D2: Moock C., "Actionscript for Flash MX: The Definitive Guide", 2nd ed., Chapter 19, O'Reilly, 2001,

but it does not rely on either of them for its reasons. The then main and 1st auxiliary requests were refused for lack of an inventive step over common knowledge alone, whereas the then 2nd auxiliary request was found to violate Articles 84 EPC 1973 and 123 (2) EPC. In an *obiter dictum* however, the 2nd auxiliary request, suitably interpreted, was also argued to lack an inventive step over common knowledge alone.

II. An appeal was filed on 6 October 2008, the appeal fee being paid on the same day. A statement of grounds of appeal was filed on 17 December 2008. It was requested that the decision be set aside and a patent be granted based on one of three sets of claims filed with the grounds of appeal.

III. With a summons to oral proceedings, the board informed the appellant about its preliminary opinion according to which the claims lacked clarity, Article 84 EPC 1973,

and the main and 1st auxiliary requests lacked an inventive step, Article 56 EPC 1973, but indicated that it was minded to remit the case to the first instance for further prosecution if a clarified 2nd auxiliary request were submitted.

IV. During oral proceedings, the appellant submitted an amended claim 1 and replaced all his earlier request with the sole request that a patent be granted based on this claim with further claims to be defined. The further application documents on file are as follows:

description pages
1, 4, 5, 8-13 as originally filed
2, 6, 7 as filed with telefax on 4 July 2007
3, 14 as filed with telefax on 4 February 2008
drawing sheets
2/4-4/4 as originally filed
1/4 as filed with telefax on 4 July 2007

V. Claim 1 reads as follows:

"A method for identifying defective program code, comprising:

providing a first program code (GoodDLL) consisting of verified program components (S1, S2, S3, S4, S5) and a second program code (NewDLL) having a plurality of modified program components (S2',S3',S5'), wherein a test of the second program code (NewDLL) failed;

(i) grouping the plurality of modified program components (S2', S3', S5') into sets,

(ii) running the test process for each of the sets of modified program components (S2', S3', S5') by creating a third program code (TestDLL) which corresponds to the second program code (NewDLL), wherein the set of modified program components (S2', S3', S5') is replaced with a set of corresponding ones of the verified program components (S2, S3, S5) and testing the third program code (TestDLL), and

(iii) determining on a test of a set failed that the set contains a defective modified program component;

automatically testing a further third program code (TestDLL), wherein in said third program code (TestDLL), one of the modified program components (S2', S3', S5') of the failed set is replaced with a corresponding one of the verified program components (S2, S3, S5) and designating the replaced modified program component as defective according to the results of the test."

VI. At the end of the oral proceedings, the chairman pronounced the decision of the board.

Reasons for the Decision

The invention

1. The application relates to the testing of software during software development. More specifically it relates to the rather common situation that a program used to work properly (*i.e.* pass the pertinent tests) at some

point in time but stopped working properly (*i.e.* failed at least some of the pertinent tests) after having been modified ("Yesterday my program worked. Today it does not. Why?"; see description, pars. 1-3). In the art, such testing of modified code is referred to as regression testing.

1.1 The application proposes a procedure based on selectively replacing modified program components by earlier, verified components - thereby "undoing" some of the modifications - and running the pertinent tests again (see fig. 2). When this second run succeeds, it is concluded that the replaced components must contain an error (see *e.g.* fig. 4). The application further proposes to perform this procedure in two stages: In the first one, an entire set of program components is replaced by corresponding set of verified ones and the program so-obtained is tested to find defective *sets* of components (see description, pars. 28-32, and fig. 3, nos. 315-335), and in the second one, individual components from defective sets are replaced and tested (see description, par. 33, and fig. 4).

1.2 Claim 1 refers to the tested program as the "first program code ... consisting of verified components", to the modified and erroneous program as the "second program code ... having ... modified program components" for which "a test ... failed", and as respective "third program code" to the programs generated during the test procedure according to the invention. Claim 1 also specifies that the modified program components of the second program code individually correspond to the "verified components" of the first one.

Article 123 (2) EPC

2. Present claim 1 is based on claim 1 of the second auxiliary request as subject to the decision under appeal.

2.1 The decision (reasons 2.15) found this earlier claim to violate Article 123 (2) EPC because, as the board understands the argument, it mentioned the replacement of individual components before the replacement of sets of components and thus in the wrong and in an undisclosed order. Claim 1 now specifies steps (ii) and (iii) in the correct order, so that this deficiency is overcome.

2.2 The application as originally filed discloses that the "changed files are ... sorted into groups" or "sorted into sets" (see par. 27) and does not literally talk about "grouping ... into sets" as does claim 1 in step (i). The application also discloses that the "sorting" could be "by any chosen criteria" such as "by the user or developer". In the board's view, the skilled person person would thus understand that the term "sorting" as used in the application is not meant to imply any order on program components, but that it is used instead to mean, more generally, "grouping". Accordingly, the board finds that the feature "grouping" is originally disclosed.

2.3 Beyond that, the board is satisfied that the wording of claim 1 finds disclosure in the application as originally filed, witness the passages of the application as cited above, and therefore conforms with Article 123 (2) EPC.

Article 84 EPC 1973

3. In the board's view, amended claim 1 is clear. In particular, the objection raised in the decision under appeal (reasons 2.16) that claim 1 of the then 2nd auxiliary request lacked clarity due to an expression lacking proper antecedent in the claim is overcome by the amended wording.

Article 56 EPC 1973

4. The decision under appeal argues that the claimed invention is obvious over common knowledge in the art of testing, especially regression testing of programs (see reasons 2.2, 2nd par. for the then main and the 1st auxiliary request, and, *obiter*, reason 3.11 for the then 2nd auxiliary request). It is argued that the "process of elimination", understood as excluding possible causes of failure, is commonly used in the field of regression testing and implies that some indispensable components have to be replaced by "trusted" components. It is also argued that the process of replacing a tested component by a "trusted" one is common practice beyond the narrow context of software testing, as is illustrated by way of an example outside of the software context. With regard to the then 2nd auxiliary request on which present claim 1 is based, the decision under appeal further argues that the "principle of divide and conquer" is common knowledge to a person skilled in the art of testing. The decision under appeal thus arrives at its conclusion that the claimed invention lacks an inventive step purely based on common knowledge in the art and without reference to any of the cited documents.

5. The board agrees with the suggestion in the decision under appeal that replacement of individual components is a commonly used strategy for locating the error in a malfunctioning system which used to work properly until a number of changes were made. For example, assume one were to modernise the wiring in a household and to replace several old (but working) components such as wire connections, safety fuses, or switches, only to find the changed wiring not to work. It is, in the board's judgment, a typical strategy to locate the cause for this error to selectively "undo" the changes, in order to see whether the wiring would work if, for instance, this old safety fuse were used instead of the new one.

5.1 The board is not convinced, however, that the claimed strategy, which the decision under appeal refers to as "divide and conquer", namely to determine defective sets of components before determining individual defective components from defective sets of components, is a commonly known testing strategy. The board considers that it depends on the nature of the system being tested whether its components can be "grouped" in a meaningful manner and in such a way that replacement of entire groups of components is possible. The board also notes that the need to speed up the testing of a large number of modifications, and thus the need for grouping, may not arise in systems in which modifications are normally made one by one and tested immediately, or in systems with a small complexity in which large numbers of modifications do not arise at all. Therefore, the board considers that the assertion that "divide and conquer" is a well-known testing strategy cannot be made independent of the system being tested.

6. As for the context of software testing, the board agrees with the decision under appeal that the person skilled in the art of testing would not hesitate to apply the "replacement" strategy to the testing of software and that, therefore, replacing an individual modified program component by an earlier, verified version to determine whether or not they are defective is obvious by direct analogy.

6.1 The board also agrees that "divide and conquer" is a well-established principle in computing, based on the idea of solving a hard problem by recursively breaking it into simpler subproblems and combining their solutions into a solution of the entire problem.

6.2 However, while the board deems the "replacement" strategy to be obvious for the testing of software almost without any appreciation of the nature and complexity of the software being tested or of nature of the tests employed and the computational costs involved, it takes the position that the relevance of "divide and conquer" in this context cannot be judged without an assessment of such aspects. The board also considers that such an assessment goes beyond general common knowledge the art of testing. The board therefore concludes that the mere existence of the well-established, but abstract, computational principle of "divide and conquer" is sufficient to establish that the claimed method of error localisation in software is obvious over the common knowledge in the art of testing.

7. The decision under appeal further asserts that elimination is also commonly used in the field of regression testing. The appellant however does not accept this

allegation without any specific evidence (grounds of appeal, p. 1, 3rd par.). The board agrees with the appellant that the allegation can reasonably be challenged and must, therefore, if relied upon be substantiated by evidence.

7.1.1 Neither D1 nor D2 provide such evidence. Although both D1 and D2 use the term "elimination" (see D1, par. 4, and D2, 3rd par. from below), they do so merely to refer to the general strategy of eliminating possible causes of an error in the process of locating the actual cause.

7.1.2 D1 talks about debugging in general terms. *Inter alia*, it discusses a game called *Clue* with the objective to deduce the solution to a crime by a "process of elimination" and it suggests that one "can do the same" in debugging by "doing numerous, carefully-selected test runs, each of which changes only one factor" and by "deduc[ing]", "[f]rom the differences in the results", "which module the error is in" and "which data structure is involved" (par. [4]). What these factors are, how they are to be changed across test runs, and how test results are to be interpreted is left open.

7.1.3 D2 discusses a methodology of debugging (p. 408 ff.) in three stages, namely "recognizing and reproducing a problem", "identifying the source of the problem" and "fixing the problem". For recognizing problems, D2 discloses the importance of testing (see p. 409, 2nd par.). For identifying the source of an error, D2 teaches to compare what "the code should be doing against what it actually is doing" and to use the "process of elimination" to narrow down the possible sources (p. 410,

lines 1-2 and 3rd par. from the bottom, line 1). The only specific illustration for such elimination in D2 is based on program tracing (see *e.g.* the *trace()* statements in the code on the top of p. 410), rather than on the modification of any "factors", let alone the replacement of program components.

7.1.4 Neither D1 nor D2 therefore provide evidence for the assertion that "elimination" is a commonly known technique in regression testing. Nor do they disclose or suggest the claimed improvement based on "divide and conquer". Hence D1 and D2 are also insufficient to establish that present claim 1 lacks an inventive step.

Summary

8. Present claim 1 was considerably amended over claim 1 of the 2nd auxiliary request as subject to the decision under appeal and overcomes the primary reasons for which the then second auxiliary request had been refused, namely those under Article 123 (2) EPC and Article 84 EPC 1973.

8.1 The board also considers that the objection under Article 56 EPC which the decision made obiter cannot be maintained because it relies on assertions about common knowledge which, in the board's judgment, either do not suffice to establish lack of inventive step of claim 1 insofar as they relate to the art of testing in general, or which must be substantiated with specific, typically written, evidence insofar as they relate to the specific field of software testing and debugging.

8.2 The board therefore decides to set aside the decision and exercises its discretion under Article 111 (1) EPC to remit the application to the department of first instance for further prosecution.

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.

2. The case is remitted to the first instance for further prosecution based on the main request (claim 1 as filed during oral proceedings, further claims to be defined).

The Registrar: The Chairman:

B. Atienza Vivancos D. H. Rees