Internal distribution code:

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

Datasheet for the decision of 15 June 2016

Case Number: T 1209/11 - 3.5.06

Application Number: 05749520.2

Publication Number: 1751650

IPC: G06F21/22

Language of the proceedings: EN

Title of invention:
ANTIVIRUS PRODUCT USING IN-KERNEL CACHE OF FILE STATE

Applicant:
Computer Associates Think, Inc.

Headword:
Antivirus product/COMPUTER ASSOCIATES THINK

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step (no)

Decisions cited:

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

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 1209/11 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 15 June 2016

Appellant: (Applicant)	Computer Associates Think, Inc. One Computer Associates Plaza Islandia, New York 11749 (US)
Representative:	Dunlop, Hugh Christopher RGC Jenkins & Co. 26 Caxton Street London SW1H 0RJ (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 2 December 2010 refusing European patent application No. 05749520.2 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Sekretaruk
Members:	M. Müller
	S. Krischer

Summary of Facts and Submissions

I. The appeal lies against the decision of the examining division dated 2 December 2010 to refuse European patent application No. 05 749 520.2 for lack of inventive step in view of

D6: WO 03/012644 A1.

II. Notice of appeal against that decision was filed on 10 February 2011, the appeal fee being paid on the same day. A statement of grounds of appeal was received on 7 April 2011. The appellant requested that the decision be set aside and that a patent be granted based on claims 1-14 as filed with letter of 16 November 2009 (as a main request) or as filed with the grounds of appeal (as an auxiliary request), the other application documents on file being:

description, pages
2-5 as published,
1, 1a, 6 as filed with letter of 11 February 2008, and
drawings, sheets
1-3 as published.

The appellant also argued that the decision was not correctly reasoned as it did not take into account all of the appellant's arguments.

III. In an annex to a summons to oral proceedings, the board informed the appellant of its preliminary opinion that the claims lacked clarity, Article 84 EPC 1973, and that the claimed subject-matter lacked inventive step over D6, Article 56 EPC 1973. The board also explained why it did not remit, at this point, the case to the department of first instance, without consideration of

its merits, for further prosecution under Article 11 RBPA.

IV. In response to the summons, with a letter dated 13 May 2016 the appellant filed two sets of amended claims 1-14 to replace the two pending sets of claims.

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

"A method of suppressing the generation of a file access event by a kernel (204) characterised by:

checking (S104, S106, S108) an in-kernel cache (206) to determine if data relating to a file is within the in-kernel cache (206);

using the data relating to the file within the in-kernel cache (206) to determine (S109) whether the file has been modified since a last time a file access event was generated for the file; and

only generating a file access event for a user-space component for the file where it is determined that the file has been modified since the last time a file access event was generated for the file."

VI. Claim 1 of the auxiliary request reads as follows:

"A method of suppressing the generation of a file access event by a kernel (204) that allows retrieval of change time and other information about a file, the method characterised by:

checking (S104, S106, S108) an in-kernel cache (206) to determine if data relating to a file is within the in-kernel cache (206);

using the data relating to the file within the in-kernel cache (206) to determine (S109) whether the file has been modified since a last time a file access event was generated for the file, by retrieving a ctime for

the file from each of the in-kernel cache and a vnode describing the file in the kernel, wherein the file is determined not to have been modified since the last time an event was generated if the ctime retrieved from the in-kernel cache is the same as the ctime retrieved from the vnode; and

only generating a file access event for a user-space component for the file where it is determined that the file has been modified since the last time a file access event was generated for the file."

VII. Oral proceedings were held as announced on 15 June 2016, at the end of which the chairman announced the decision of the board.

Reasons for the Decision

Article 11 RPBA

1. The appellant points out that its letter dated 16 November 2009 gave *three* reasons for disagreeing with the examining division's line of argument (see that letter, page 2, paragraphs 2-6), only two of which were addressed in the decision (see reasons 4: "two main arguments"), and argues that this makes the decision insufficiently reasoned (see grounds of appeal 1.22).

1.1 The third reason, allegedly disregarded by the examining division, concerned the question whether, even accepting that it was obvious to include some functionality in the operating system kernel, it was also obvious for the skilled person to include precisely the functionality claimed.

1.2 On a literal reading, it may indeed appear that the examining division did not address this "third reason" in connection with the main request. However, the reference to "two *main* arguments" in the decision (reasons 4, first two lines; emphasis by the board) may also suggest that the examining division merely considered the third argument to be of lesser importance. Moreover, the examining division has at least stated its opinion that the skilled person would, "guided by common-knowledge", place "time-critical elements in the kernel" according to the circumstances and without the need to exercise inventive skill (see reasons 5.2). The board thus has no reason to suspect that the examining division disregarded the appellant's "third" argument.

1.3 Moreover, the board cannot see what purpose a remittal would serve even if for the sake of argument the alleged incompleteness in the decision were accepted and regarded as a fundamental deficiency in the sense of Article 11 RPBA. The board considers that the inventive-step objection itself as laid out in the decision (see its reasons 2-2.6) is sufficiently reasoned and clearly comprehensible, and that this fact constitutes at least a special reason within the meaning of Article 11 RPBA for not remitting the case.

1.4 The board therefore decided not to remit the case at hand to the first instance for further prosecution, without assessing its substantive merits.

The invention

2. The application is generally concerned with the efficiency of anti-virus checking and, more

specifically, with reducing the overhead caused by on-access virus scanning.

2.1 It is stated to be known (page 1, lines 16-18) that anti-virus products may "include a kernel-level component that intercepts an attempt to access a file and passes information about that attempt as an event to a user-space component that makes a decision whether the file has to be scanned for viruses". And it is observed that known systems generate superfluous "events" in situations where a "file has been previously scanned and it is known to be safe to let the operation proceed" (lines 22-23). Antivirus checking following such a superfluous event causes unnecessary delays.

2.2 To solve this problem, the invention proposes not to generate a file access "event" if it can be established that the file has not changed since the last event was generated (see the main request).

2.3 The application makes specific reference to "Unix or Unix-like operating systems" (see page 3, lines 13-14; page 4, lines 7-9). It is disclosed that in such systems, files may be maintained in on-disk objects called "inodes", that such "inodes" have a "ctime" parameter indicating the last time the file was changed, and that the "ctime" is exclusively changed by the file system (page 3, lines 13-20). Thus, "it is possible to use ctime of a file as an indicator of when that file has changed" (lines 20-22). Furthermore, "vnodes" are referred to which act as "pointers" into an "in-kernel cache" of file data and are "persistent for the lifetime of a file", i.e. as long as references to the file exist (see page 4, lines 3-10).

2.4 In this context, it is disclosed that the "ctime for [a] file from the cache and vnode" are compared and equality is taken to mean that "the file has not changed since the time it was added to the cache", in which case "no event need be generated for that file" (see page 4, lines 23-26, and the auxiliary request).

Claim construction

3. The wording of the claims suffers from a number of deficiencies. When this was addressed in the summons and during oral proceedings, the appellant declared its willingness to file suitable clarifying amendments. In view of this fact, the board decided to leave open the question whether the claims as they stand lack clarity in the sense of Article 84 EPC 1973 - and could be clarified within the limits of Article 123(2) EPC - and to construe the claims in the light of the description and decide the case based on an assessment of inventive step.

4. The claims of both requests relate to "suppressing the generation of a file access event by a kernel" and to "only generating a file access for a user-space component where it is determined that the file has been modified since the last time a file access event was generated for the file".

4.1 This language implies that the file access event *is not generated* ("suppressing the generation") under the given condition. This is also disclosed in the passage of the description (page 4, lines 24-29) which the appellant indicated as the basis for the formulation used: if the time stamps are the same "then the file has not changed [...] and no event need be generated". On the other hand, the literal claim language ("only")

arguably leaves open whether an event *is actually generated* when the time stamps differ. In the board's view, however, this is implied by the description, which states that the invention is an improvement over background art which generated an event for *every* file access (see page 1, lines 16-18).

4.2 The invention proposes to suppress file access events if it can be established - by two equal of time stamps - that the file has not been modified (see the description, page 4, lines 24-29). However, the opposite is not true: even if the time stamps differ, the file may not have been modified.

4.3 Accordingly, the board takes the claims to specify that a "file access event for a user-space component" *is actually generated* unless it can be excluded, based on the time-stamp information, that the file has been modified.

5. A central feature of the claimed invention is that the "events" in questions are generated "by a kernel" rather than in user space. The kernel is a fundamental, low-level component of an operating system which interfaces with the CPU and other electronic components of a computer. Beyond that, however, it is not implicit in the notion of a "kernel" precisely what functionality it contains. For instance, there are "microkernels" which contain only what is absolutely necessary, but also large "monolithic" kernels, and then there are "hybrid kernels" where users get to pick and choose what they want to go into the kernel. Both monolithic kernels and microkernels exist in the Unix world.

6. The claims of the auxiliary request refer to "a ctime for a file" and to "a vnode describing the file in the kernel". The board takes the view that the skilled person would understand these terms to imply a "Unix and Unix-like operating systems" (see the description, page 3, lines 14-15). However, it would appear to the board that the specific meaning of the ctime identifier and the vnode concept may differ even across members of the Unix family. Also, to the best of the board's knowledge (and in contrast to the claim language), the vnodes do not have a field called "ctime" - although they do allow access to the time that the vnode was last changed.

6.1 For the purpose of assessing inventive step, therefore, the board interprets the claims as follows.

6.2 The claimed "in-kernel cache" is a cache for "on disk object[s] called inodes" (see e.g. description page 3, lines 15-16, and original claim 39) which contain "file data" such as a time stamp (which happens to be called "ctime") which indicates the last time the file was changed on disk. Whenever a file is accessed, it will be checked whether the inode cache contains "data relating to [this] file". If not, the inode of the accessed file will be cached at that point.

6.3 The vnodes are file system objects which, *inter alia*, point to inodes (see page 4, lines 6-7). File accesses immediately affect the vnodes. Modifications to the file will eventually be written to disk, in particular when the file is closed, and only then affect the inode data. In the meantime the time stamp of the vnode and its corresponding inode may differ. In the board's understanding, therefore, the vnode time stamp will

either be more recent than or identical to the inode time stamp.

6.4 This understanding is consistent with the description and was not challenged by the appellant's representative during the oral proceedings.

The prior art

7. D6 is concerned with anti-malware scanning for mobile devices (see pages 1 and 2), *inter alia* with on-access scanning (see page 10, last item in the table; page 13, table 1D; page 20 et seq.).

7.1 The on-access scanner is a user-space application said to be "governed by operating-system hooks [...] which provide document access notification". It is thus "notified of various events relating to accessing documents" and then "determines whether the document needs to be scanned" (see page 20, lines 26-27, 39-40).

7.2 It is stated in D6 that "For a comprehensive protection it is crucial to be able to hook into all file access events prior to them happening and after they have occurred". The specific events in question are "file create", "file open" or "execute program", "file close" and "file rename" (see pages 21-22).

7.3 It is disclosed that files must be checked when they are opened, and that they may be checked when they are closed, provided that changes were made to the file. Whether this is the case is determined by comparing "identification information" obtained from the file when it was opened with "this same information" when the file is about to be closed (page 22, lines 9-24).

The nature of the "identification information" is not specified.

7.4 During oral proceedings, the appellant suggested that the language used in D6 ("obtained from it") implied that the identification information had to be obtained from the *contents* of the file rather than from its *metadata*, such as that contained in the inode. The board does not agree that D6 allows this conclusion to be drawn. Rather, the manner in which D6 refers to the "file" includes, in the board's view, the file metadata as well.

Inventive step

8. The appellant and the board agree that D6 is a suitable starting point for the assessment of inventive step.

8.1 D6 discloses an on-access malware scanner which acts in response to the "events" generated by the operating system when an "attempt" is made to access a file. As the board understands D6, these events are generated unconditionally and are not meant to be "suppressed" (see D6, page 21, lines 16-18). In this respect, the board agrees with the appellant (see grounds of appeal, point 1.12).

8.2 However, the "events" of D6 do not correspond to the "events" as claimed but rather to the "attempts" to access a file. These "attempts" are not meant to be suppressed either; rather, the board understands the application to assume that they are unconditionally generated. What are referred to as "events" in the claims are "signals" to the antivirus scanner in user

space (page 1, lines 16-21; page 5, paragraph 1), and it is only these signals which may be suppressed.

8.3 D6 discloses that in response to certain file access events the anti-malware scanner may or may not have to perform certain actions. Specifically, it is disclosed that, whenever a file is closed (i.e. when a file close event is generated), the anti-malware scanner may have to perform a "more resource intensive task" if changes were made to the file while it was open (see page 22, lines 9-18).

8.4 The board takes the view that the claimed "events" correspond in D6 to the control "signals" which instruct the anti-malware scanner to scan a file. From this perspective, D6 discloses that the event may be suppressed if the file in question has not changed.

8.5 Furthermore, it is the board's understanding of D6 that such a "signal" is invariably generated whenever a file is opened. When that file is closed, therefore, establishing whether the file was changed while it was open is normally equivalent to determining whether the file has been changed "since the last time" an instruction signal to the anti-malware scanner "was generated for the file".

9. The board thus considers that, in contrast to claim 1 of the main request, D6 does not disclose:

i) an "in-kernel cache", let alone "checking [...] an in-kernel cache [...] to determine if data relating to a file is within the in-kernel cache",

ii) that the decision to "suppress" (i.e. not to generate) an instruction is made based on data contained in that "in-kernel cache", and

iii) that the instructions ("events") controlling the anti-malware scanner may be suppressed "by [the] kernel" (and hence are otherwise generated).

In contrast to claim 1 of the auxiliary request, D6 also does not disclose

iv) the comparison between "a ctime for the file from each of the in-kernel cache and a vnode describing the file in the kernel".

9.1 The board takes the view that these differences address the problem of implementing the system of D6, especially as regards choosing which "identification information" is to be used to determine whether the file has been changed, with a view to increasing execution speed.

9.2 *Re i)* D6 discloses that the anti-malware scanner is meant to run on a variety of operating systems (page 2, last paragraph). Even though no member of the Unix family is specifically disclosed in D6, the board considers it to be obvious that the scanner may have to be deployed on, say, a Linux system (the board notes in passing that the Android operating system for mobile devices is based on Linux and was available before the priority date of the present application). In Unix-based operating systems that may obviously be chosen, the claimed "in-kernel cache" (i.e. the inode cache), the vnodes and the claimed time stamps are available.

9.3 *Re ii) and iv)* The "identification information" mentioned in D6 must be such as to allow a decision whether or not the file has been (or rather: may have been) changed. The board considers that it would have been a natural - and hence obvious - choice for the skilled person to use available time stamps indicating when a file was "changed". Since such time stamps exist for the very purpose of recording information about file change events, it is the board's conviction that the skilled person not only could have but indeed would have made this choice. Moreover, in a Unix context it would have been obvious for the skilled person to pick the time stamps specifically claimed.

9.4 With regard to *difference iv)* the appellant argues that the ctime parameter has the additional advantage of being particularly secure. In this regard, the appellant relies on the disclosure of the application on page 3 (lines 18-20). The board does not accept this argument. Firstly, the board notes that the application on page 3 refers only to the ctime of the *inodes* in the in-kernel cache but not to the change time parameter of the *vnodes,* and thus seems unable to support the appellant's argument. Secondly, if the choice of certain parameters is obvious - as the board thinks it is - it does not become less obvious if the way in which the parameters are made available have a security advantage. In other words, the allegedly improved security is, in the board's view, at best a bonus effect of the otherwise obvious choice of "identification parameters", and has no bearing on the question of inventive step.

9.5 *Re iii)* The question is whether it would have been obvious for the skilled person starting from D6 to make

the decision to "suppress" (i.e. not to generate) a control signal in kernel space or in user space.

9.5.1 The appellant states that not all "problems related to improving the speed of computer programs (and improving the speed of file access in particular) can be solved simply by including additional functionality in the kernel" (see grounds of appeal, point 1.19). In general, the board agrees with this statement.

9.5.2 In the specific case at hand, however, the speed-up is considered realistic by the board. If a decision is to be made *based on information which is directly available in the kernel*, it is plausible to assume that it would be quicker to make that decision *in kernel space* rather than to access the information from user space and make the decision there.

9.5.3 The board agrees with the appellant that the decision on what goes or does not go into the kernel is not, *per se*, a business decision - even if the decision may well be influenced by non-technical criteria. The examining division too found this decision to only be "*also* a business decision" (see reasons 2.6, emphasis by the board).

9.5.4 The board considers that the design of the operating system kernel is primarily a matter of programming. It is known, for instance, that monolithic kernels are easier to develop but are less flexible and require more memory space. The skilled person would therefore weigh up issues such as ease of development and maintenance against memory requirements.

9.5.5 Moreover, the board cannot see any particular technical difficulty that might discourage the skilled person

from providing kernel support for a particular decision that has to be made anyway.

9.6 In summary, the board finds that the feature according to which the decision (whether or not to generate an "event", i.e. a signal into user space) is made *by a kernel* would have been obvious for the skilled person in view of D6.

10. As a consequence of the above, the board comes to the conclusion that claim 1 of both the main and the auxiliary request lacks inventive step over D6, Article 56 EPC 1973.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

A. Vottner W. Sekretaruk

Decision electronically authenticated