

The opinion in support of the decision being entered today
is not binding precedent of the Board.

Paper No. 26

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte MICHAEL A. PERKS

Appeal No. 2000-1238
Application 08/566,638

ON BRIEF

Before MARTIN, DIXON, and GROSS, Administrative Patent Judges.
MARTIN, Administrative Patent Judge.

DECISION ON APPEAL

This is an appeal from the final rejection of claims
1-14 over prior art.¹ We reverse.

¹ As noted in the "Status of Amendments After Final" section in Appellant's brief (at 3), which section the examiner has indicated is correct (Answer at 1), an amendment (Paper No. 20) was filed on February 26, 1999, canceling claims 15-18. This amendment has not yet been formally entered.

The invention

The invention is a semaphore manager data structure for managing semaphores in a multi-tasking computer system having a storage means. Figure 7 shows five different ways to allocate semaphores to classes and objects in an object-oriented computing system environment. The rounded squares 70 represent Classes A, B, and C. Each circle 79 represents an object and the number in each circle is the corresponding semaphore. The "Global Semaphore" allocation technique assigns the same semaphore (i.e., Semaphore 1) to every object in all three classes. The "Semaphore per Group of Classes" technique assigns Semaphore 1 to all of the objects in classes A and C and Semaphore 2 to all of the objects in Class B. The "Semaphore per Class" technique assigns Semaphore 1 to all of the objects in Class A, Semaphore 2 to all of the objects in Class B, and Semaphore 3 to all of the objects in Class C. The "Semaphore per Group of Objects" technique assigns Semaphore 1 to the first object in every class, Semaphore 2 to the second object in every class, and so on, while the "Semaphore per Object" technique assigns a unique semaphore to every object. The specification explains that these allocation schemes

can be viewed in terms of a granularity scale, with the single global semaphore method as the least granular and the one semaphore per object method as the most granular. The rest of the allocation schemes lie somewhere in the middle of the granularity scale. Incidentally, the most desirable allocation scheme would consist of not too many semaphores such that deadlock may become a problem and not too few semaphores such that concurrency may be limited (or no concurrency at all as in the case of the single global semaphore method).

Specification at 4, ll. 1-9. Appellant's solution is to employ a semaphore manager data structure which implements the following assignment criteria: "a class can only be assigned to one semaphore but a semaphore can be assigned to more than one class" (id. at 5, ll. 19-21). The advantages of this data structure

include the number of actual semaphores that can be controlled so that the computer system would not be overwhelmed, the semaphore mapping can be performed statically at the time of compilation or dynamically during time of execution, potential deadlock situations can be reduced due to the assurance that only one class can request one semaphore at a time, and the semaphore tracing or debugging capabilities can be enhanced because all semaphores are managed centrally.

Id. at 5, ll. 21-28. In addition to classes which include objects, Appellant's preferred embodiment employs a semaphore class and a semaphore manager class. Id. at 10, ll. 18-21.

A specific example, including sections 1-6, is given in the specification at page 11, line 23 to page 12, line 33. In section 1, each of the five classes is assigned a unique index

number 1-5, with index number 0 being specifically reserved for the semaphore manager. In sections 3 and 4, the semaphore utilized by the semaphore manager is set to 0 and the remaining three available semaphores are numbered 1 to 3, with semaphore 1 being assigned to MyClass 3, semaphore 2 being assigned to MyClasses 1, 2, and 5, and semaphore 3 being assigned to MyClass 4.

The claims

Claim 1 is representative:

1. A semaphore manager data structure for managing semaphores in a multi-tasking computer system having a storage means, said data structure comprising:

a plurality of indices residing in said storage means, wherein each of said plurality of indices defines a corresponding class;

a plurality of semaphore numbers residing in said storage means, wherein each of said plurality of semaphore numbers defines a corresponding semaphore;
and

a mapping table residing in said storage means, wherein said mapping table defines an assignment of each of said semaphores to each of said classes by utilizing said plurality of indices and said plurality of semaphore numbers, wherein a class can be assigned to only one semaphore and said semaphore may be concurrently assigned to more than one class.

The examiner's rejection

The examiner's rejections are based on the following patent and publications:

Holt et al. (Holt) 5,394,551 Feb. 28, 1995

Grady Booch, Object-Oriented Analysis and Design 88, 89, and 360-65 (Addison-Wesley Publishing Company 1994) (Booch)

D. Decouchant et al., A Synchronization Mechanism for an Object Oriented Distributed System, 152-59 (IEEE 1991) (Decouchant)

Claims 1, 3, and 4 stand rejected under § 103(a) for obviousness over Holt in view of Decouchant and Booch.

Referring to Holt's Figure 1, Holt discloses using semaphores in a data processing system having a plurality of processing nodes which are interconnected by a communication network and have access to shared resources, such as a shared memory (col. 1, ll. 9-13). Figure 2 shows that each node includes a processing unit 20 which has access to access shared resources, such as areas of a shared memory (col. 2, ll. 16-42-58). Furthermore,

[e]ach node includes a semaphore unit 22 which controls access to the shared resources 21, using a set of semaphore locations 23, a semaphore ownership table 24 and a semaphore queue 25. Each node has its own local copies of the semaphore locations and the semaphore ownership table, and has its own semaphore queue.

Column 2, ll. 59-65. Each shared resource has a particular semaphore location associated with it (col. 2, ll. 66-67). That is, a different semaphore location is assigned to each shared

resource. "The semaphore ownership table 24 consists of a number of sections, one for each node. Each section has a fixed number of slots, each of which can hold an entry, defining the ownership state of a particular semaphore." Column 3, ll. 19-22.

Specifically, each entry in the ownership table includes the following two fields: (1) ADDRESS, which is the virtual address of the semaphore location to which the entry relates; and (2) STATE, which is the ownership state of the semaphore location (col. 3, ll. 23-28). The meanings of the ownership states are defined as follows:

IDLE: the semaphore is not owned by any node.
OWNED: the semaphore is owned by the local node.
DISCARD: ownership of the semaphore has been relinquished.
OTHER-OWNED: the semaphore is owned by a remote node.
QUEUED: the semaphore has one or more suspended semaphore operations in the queue 25.

Column 3, ll. 39-47. It is evident from these ownership state definitions and the abstract, reproduced in part below, that a semaphore cannot be concurrently owned by plural nodes:

When a node requires a semaphore operation on a particular semaphore, a semaphore message is broadcast to all the nodes instructing them to perform the semaphore operation on their local copies of the semaphore. If the semaphore is unowned, the node must suspend the semaphore operation until the message returns, so as to ensure correct chronology for the semaphore operation. If, however, the semaphore [is] owned by this node, the node can perform the semaphore operation without waiting for the message to return. This speeds up the semaphore mechanism. If the

semaphore is owned by another node, that other node relinquishes ownership so that the semaphore operation can be performed.

(Emphasis added.)

The examiner's case for obviousness is stated as follows:

Holt et al[.] refers to nodes and does not teach classes. Decouchant et al[.,] however, shows a semaphore which can be concurrently assigned to more than one class in an object-oriented environment (subclassing and overloading, section 5 synchronization and inheritance.)

It would have been obvious to one of ordinary skill in the art to provide semaphores for classes since such semaphores reduce the number of processes and allow efficient blocking of other objects.

(Bolding omitted.) Paper No. 13, at 3. The remarks in the Answer suggest the examiner is proposing to modify Holt's system so as to employ object-oriented programming at the various nodes, to assign object-oriented semaphores of the type taught by Decouchant and Booch to single classes and multiple classes, and to store the assignment information for the object-oriented semaphores in Holt's semaphore ownership table 24, which also stores ownership information about Holt's semaphores that are associated with the shared resources. Thus, the examiner reads claim 1 on Holt as modified in the following manner:

Holt et al. shows a plurality of indices (section for each node, col. 3 lines 19-20) residing in said storage means, wherein each of said plurality of indices defines a class . . . ;

a plurality of semaphore numbers (virtual address of semaphore, col. 3 lines 26-27) residing in said storage means, wherein each of said plurality of semaphore numbers defines a corresponding semaphore; and

a mapping table residing in said storage means (semaphore ownership table, col. 3 line 19), wherein said mapping table defines an assignment of each of said semaphores to each of said classes (semaphore table defines the assignment of nodes to semaphores, col. [3, lines] 19-46) by utilizing said plurality of indices and said plurality of semaphore numbers and a semaphore can be assigned to more than one class (a semaphore can be owned by different nodes, col. 3[,] lines 40-45.

(Bolding omitted.) Paper No. 13, at 3.

We agree with Appellant that the examiner has failed to establish a prima facie case of obviousness. The examiner has not adequately explained, and it is not otherwise apparent to us, why one skilled in the art, absent the guidance provided by Appellant's disclosure and claims, would have been motivated to (1) employ object-oriented programming in Holt's computing system, (2) employ Decouchant's and Booth's object-oriented semaphore techniques in Holt's system thus modified, and (3) store the assignments of the object-oriented semaphores in Holt's semaphore ownership table. Even assuming for the sake of argument that it is physically possible to combine the reference teachings in the manner proposed by the examiner, that is an insufficient basis for combining their teachings in the manner

Appeal No. 2000-1238
Application 08/566,638

proposed by the examiner or any other manner. See In re Kotzab, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1316 (Fed. Cir. 2000) ("to establish obviousness based on a combination of the elements disclosed in the prior art, there must be some motivation, suggestion or teaching of the desirability of making the specific combination that was made by the applicant. See In re Dance, 160 F.3d 1339, 1343, 48 USPQ2d 1635, 1637 (Fed. Cir. 1998).").

Appeal No. 2000-1238
Application 08/566,638

Because the examiner has failed to establish the obviousness of combining the reference teachings in the proposed manner, the rejection of claims 1-14 is reversed.

REVERSED

JOHN C. MARTIN)	
Administrative Patent Judge)	
)	
)	
JOSEPH L. DIXON)	BOARD OF PATENT
Administrative Patent Judge)	APPEALS AND
)	INTERFERENCES
)	
)	
ANITA PELLMAN GROSS)	
Administrative Patent Judge)	

Appeal No. 2000-1238
Application 08/566,638

cc:

Bracewell & Patterson, L.L.P. #25
Intellectual Property Law
P.O. Box 969
Austin, TX 78767-0969