

THIS OPINION WAS NOT WRITTEN FOR PUBLICATION

The opinion in support of the decision being entered today
(1) was not written for publication in a law journal and
(2) is not binding precedent of the Board.

Paper No. 11

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte JEAN BERTHE and HERVE PERRINOT

Appeal No. 1997-1009
Application Serial No. 08/033,731¹

ON BRIEF

Before THOMAS, MARTIN, and LEE, Administrative Patent Judges.
MARTIN, Administrative Patent Judge.

DECISION ON APPEAL

This is an appeal under 35 U.S.C. § 134 from the
examiner's final rejection of claims 1-11, all of the pending
claims, under 35 U.S.C. § 103. We reverse.

The invention

The invention is directed to a method for capturing data
for debugging purposes when an exception condition in the form
of a "Major Error," occurs during operation of a software

¹ Application for patent filed March 18, 1993.

program (Spec. at 1, lines 4-7 and p. 7, lines 16-20). When a "Major Error" occurs, "the program loses control of the operations and is no longer able to detect itself the error or failure" (Spec. at 6, lines 24-25).

Appellants' invention can be summarized as follows. The program code is divided into logical subsets called Tasks (Spec. at 13, lines 6-8). Each Task or subset is used to develop a corresponding data table including a plurality of Families each containing a description of the data fields to capture in the event of the corresponding error (Spec. at 13, lines 20-25). In a steady state, the Families are activated one after the other at the key points of the program code such that at each key point a Family is selected by the Task in the Activation Table (Spec. at 18, lines 5-10). When a Major Error occurs, the task loses control of the operations and the error is instead detected by a control program, which calls on an error handler program to retrieve the data fields identified in the last subset (of the data table) that was selected by the activation table (Spec. at 18, lines 18-21).

The claims

Claims 1 and 9 are the only independent claims. Claim 1, which is representative, reads as follows:

1. A selective method for capturing data in software exception conditions (Major Errors) during the operation of a data processing system, said system operating with at least one task, each said task being endowed with a dedicated memory space and being executed on instruction of a control program, characterized in that it involves the steps of:

- defining dynamically for each task a data table (Task Data Table 200), said data table being divided into a certain number of subsets (Families 206)
- describing in each subset (Family 206) once at the beginning of the task execution, the data fields, permanently defined in the memory of the task, which are relevant for the Major Errors anticipated by the task,
- describing in each subset (Family 206), in the course of the execution of the task, the data fields, dynamically defined by means of temporary memory allocation, which are relevant for the exception condition (Major Error) anticipated by the task,
- selecting (403, 703) at each potential exception condition in the code the appropriate subset (Family 206) in an activation table (405 or 705) unique for each task,
- detecting, when it occurs, an exception condition (409, 709) and identifying the faulty task,
- transferring the control of the operations to an error handler code (Error Handler 411 or 711), said code being endowed with a priority level higher than the level of the tasks and being authorized to access the tasks memory,
- retrieving by the error handler code (Error Handler 411 or 711) the pertinent data fields from the descriptions (208)

However, we agree with appellants that Cobb does not disclose the claimed steps of "selecting . . . at each potential exception condition in the code the appropriate subset . . . in an activation table . . . unique for each task" (emphasis added), and then, in response to detection of the occurrence of an exception condition, "retrieving . . . the pertinent data fields from the descriptions . . . contained in the last subset . . . selected in the activation table . . . associated with the faulty task" (emphasis added). These limitations make it clear that the activation table, throughout execution of the program code, identifies the subset in the data table which corresponds to the potential exception condition currently of concern. Cobb, in contrast, does not track potential errors or the corresponding information in the data table. Instead, Cobb waits until an actual error has been detected to determine which data in the data table corresponds thereto:

The Early Detection Data Capture process uses permanently placed error detection points located strategically within a software program when initially developed. The detection points check the status of the software program throughout its execution. If an error is detected, the EDDC process is called. Unless an error is detected, the

EDDC process remains completely inactive. [Col. 3,
lines 31-38.]

The examiner's argument that the foregoing claim limitations are satisfied because "figure 5, for example teaches selecting causes for potential exceptions (see also Figure 6-9)" (Answer at 10) is unpersuasive, because none of these figures relate to identifying, during execution of the program code, the data in the data table which corresponds to the potential error currently of concern. Consequently, we are not persuaded that Cobb, the only reference before us, discloses or suggests the "selecting" and "retrieving" steps of claim 1 or the corresponding steps in claim 9, the only other independent claim.

Nor are we persuaded that Cobb discloses or suggests claim 1's step of "describing in each subset . . . , in the course of the execution of the task, the data fields, dynamically defined by means of a temporary memory allocation, which are relevant for the exception condition (Major Error) anticipated by the task."² The examiner contends this limitation is satisfied because "dynamic memory allocation is

² No such limitation appears in claims 9-11.

inherent to any process operating on a computer, [sic, ;] when that process needs more space the computer's operating system allocates that memory for its temporary usage" (Answer at 4). While this is true, it does not satisfy the claim language in question, which requires that the temporary memory locations be stored in the corresponding subsets of the data table.

For the foregoing reasons, the rejection of claim 1 and its dependent claims 2-8 is reversed, as is the rejection of independent claim 9, and its dependent claims 10 and 11.

REVERSED

JAMES D. THOMAS)	
Administrative Patent Judge)	
)	
)	
)	BOARD OF PATENT
JOHN C. MARTIN)	
Administrative Patent Judge)	APPEALS AND
)	
)	INTERFERENCES
)	
JAMESON LEE)	
Administrative Patent Judge)	

Appeal No. 1997-1009
Application 08/033,731

JCM:lmb

EDWARD H. DUFFIELD
IBM CORP.
DEPT. 972/BLDG. 205
P.O. BOX 12195
RESEARCH TRIANGLE PARK, NC 27709

Appeal No. 1997-1009
Application 08/033,731