Apple, Inc. v. Motorola, Inc. et al Doc. 97 Att. 15

EXHIBIT 15

Dockets.Justia.com

#16 C
7-5-00
P12

Docket No. 3048-7035US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s):	Orton et al.	OTLC Docket:	P-046.63
Serial No.:	09/140,523	Group Art Unit:	2762
Filed:	August 26, 1998	Examiner:	J. Chavis
For:	OBJECT-ORIENTED OPERATING SYSTEM		

RECEIVED
JUL -5 2000
TC 2700 MAIL ROOM

OIPE
JUN 29 2000
PATENT & TRADEMARK OFFICE

PRELIMINARY AMENDMENT

Honorable Assistant Commissioner
of Patents and Trademarks
Washington, D.C. 20231
Sir:

In response to the Office Action of February 1, 2000, please amend the application as follows:

IN THE CLAIMS:

Please AMEND the claims as follows:

Please cancel claim 11 without prejudice.

Please amend claim 12 as follows:

2 ~~12.~~ (AMENDED) The computer system of claim [11] ~~10~~ 1, wherein the procedural program logic code further comprises:

C1

procedural program logic code portions specific to each object-oriented method to issue one or ore procedural function calls compatible with the native interface to control the native system services performed by the hardware environment to correspond to the native system services required by the object-oriented method.

Please cancel claims 13 - 15 without prejudice.

Please amend claim 16 as follows:

16. (Amended) A computer system, comprising:

computer hardware for performing native system services;

a procedural operating system, having a native interface, for controlling the computer hardware to perform the native system services;

object oriented methods requiring native system services;

procedural program logic code, responsive to invocations of the object-oriented methods, for causing the procedural operating system to control the computer hardware to perform the required native system services;

[The computer system of claim 11, further comprising:]

executable program memory associate with the computer hardware for runtime execution of the procedural operating system, invocations of the object-oriented methods and related portions of the procedural program logic code;

means for making determinations during runtime execution if object-oriented methods to be invoked are present in the executable program memory; and

a runtime loader, responsive to the determinations, to selectively load required object-oriented methods into the executable program memory during runtime before invocation of the object-oriented methods.

Please cancel claim 21 without prejudice.

Please amend claim 22 as follows:

22. (AMENDED) A method for operating a computer system, comprising the steps of:

executing a procedural operating system on computer hardware, the procedural operating system including a native interface, responsive to procedural function calls, for providing native system services;

issuing calls, compatible with the native interface, to provide the native system services in response to invocations of object-oriented methods requiring such native system services;

[The method of claim 21, further comprising the steps of:]

determining if object-oriented methods to be invoked during runtime execution are present in executable program memory associated with the computer hardware; and

selectively loading the object-oriented methods into the executable program memory during runtime before invocation thereof, if not yet loaded.

Please amend claim 25 as follows:

25. (AMENDED) The method of claim [21] 22, wherein the step of issuing calls, compatible with the native interface, to provide the native system services in response to invocations of object-oriented methods requiring such native system services, further comprises the step of:

adapting the native services provided by the procedural operating system to be compatible with the native system services required by the associated object-oriented method.

Please amend claim 27 as follows:

27. (AMENDED) A method for operating a computer system, comprising the steps of:

executing a procedural operating system, based on Windows or Unix operating systems, on a Unix or IBM compatible computer hardware environment;

providing an object-oriented interface, executing on the computer hardware environment, and responsive to object-oriented programming, for instantiating objects from object-oriented classes, encapsulating data for exclusive use with each object, and invoking object-oriented methods in the objects for operating on the encapsulated data; [and]

providing procedural programming logic code, responsive to selected ones of said invoked object-oriented methods requiring native system services, for issuing procedural calls, compatible with a native interface of the procedural operating system, to cause the hardware environment to provide the native system services in response to the object-oriented methods[.] ;

and

loading the methods during runtime before invocation thereof;

whereby a choice of which system implementation to use can be deferred to run-time.

REMARKS

Status Of The Claims

Claims 11, 13 - 15, 21 are cancelled without prejudice.

Claims 12, 16-20, 22-30, as amended, remain in the case remain in the case.

Rejection Of The Claims Over The Prior Art

Claims 11-30 are rejected under 35 U.S.C §103 as being unpatentable over the Schmidt, "Systems Programming With C++ Wrappers", C++ Report, October 1992.

Applicants' Response

In response, the Applicant states that Schmidt fails to disclose or suggest loading the method during runtime before invocation thereof, as claimed by the Applicant. The examiner cites the Schmidt reference to show that the applications built using Schmidt's libraries ultimately make system calls at run-time. This is true of all applications, not merely Schmidt's. Ultimately, the actual calls on system functionality get made at runtime in all cases. What the Applicants are claiming is that the claimed invention can defer the decision about which system implementation to use until run time. The Applicant's claimed invention loads the method during runtime just before invocation thereof. Adding the claim element of "loading the method during runtime before invocation thereof" means that (unlike Schmidt) it is possible to wait until the program is running before the particular library is chosen and used by the program. There is nothing in Schmidt that even suggests this claimed feature.

In the claimed invention, the application can be written and compiled, and only when it is actually running does the particular library get linked to it to specify which actual code (including the code with system calls specific to this platform) would be used. In the case of Schmidt, the developer makes the decision which library to use at development time, not run time. Schmidt then specifies a particular library with which to link (still at development time) and the resulting application is now hard-coded to work on only one particular system (and then, of course, the actual system calls eventually occur at run-time). Thus, in the claimed invention, the choice of which system implementation to use can be deferred to run-time, whereas in Schmidt's disclosed system it is determined prior to run-time and once determined can no longer be changed at run time".

Schmidt's repeated references to using "inline functions" is proof that his the technique is limited to compile-time. Schmidt also refers repeatedly to the benefit of his technique as allowing error checking at "compile-time" rather than "run-time". He also makes repeated references to "stronger type-checking", "type-safe operations", and "type mismatch detection" which are compile time features. Schmidt also talks about developers "designing and implementing" their own "wrappers", stating it "is an effective way for programmers to learn C++". Schmidt fails to disclose running or executing at run-time, as claimed by the Applicant.

The Applicant's claimed invention allows writing a program once, compiling it once, and simply by the differences in the platform on which it runs, use the appropriate wrapper. Schmidt does provide for different system implementations for the same programming API, but he requires that the choice be specified by the developer, rather than waiting until it is determined upon which system the program is actually running. Thus in claimed invention, the object oriented statements using a wrapper are located by the system at run-time while running or executing, whereas in Schmidt the locating is completed at development time. Schmidt fails to disclose or even suggest the Applicant's claimed invention.

By the above remarks, the Applicant believes all of the issues raise by the Examiner have been resolved. Accordingly, the Applicant respectfully requests the Examiner's reconsideration of the claims, allow the claims and pass the case to issue.

The Assistant Commissioner is hereby authorized to charge any additional fees which may be required for the timely consideration of this amendment under 37 C.F.R. §§ 1.16 , or credit any overpayment to Deposit Account No. 13-4503, Order No. 3048-7035.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: 6/28/2000

By:

John E. Hoel
Registration No. 26,279
202-857-7887 - Telephone
202-857-7929 - Facsimile

SENDER'S ADDRESS:
Morgan & Finnegan L.L.P.
1775 Eye Street, N.W. Suite 400
Washington, D.C. 20006