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 July 2013

Case Number:	T 0608/09 - 3.5.06
Application Number:	02077625.8
Publication Number:	1306756
IPC:	G06F 9/445, G06F 9/46
Language of the proceedings:	EN

Title of invention:
Data processing method and device

Applicant:
Sony Corporation

Headword:
Object migration/SONY

Relevant legal provisions (EPC 1973):
EPC Art. 56, 87 (1), 89

Keyword:
"Inventive step - yes"

Decisions cited:
-

Catchword:
-

This datasheet is not part of the Decision.
It can be changed at any time and without notice.

Case Number: T 0608/09 - 3.5.06

DECISION
of the Technical Board of Appeal 3.5.06
of 9 July 2013

Appellant: (Applicant)	Sony Corporation 1-7-1 Konan Minato-ku Tokyo (JP)
Representative:	Turner, James Arthur D Young & Co LLP 120 Holborn London EC1N 2DY (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted 16 October 2008 refusing European patent application No. 02077625.8 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman:	D. H. Rees
Members:	M. Müller W. Sekretaruk

Summary of Facts and Submissions

I. The appeal lies against the decision of the examining division dated 16 October 2008 to refuse European patent application 02077625.8 for not complying with Article 123 (2) EPC and for lack of an inventive step, Article 56 EPC 1973, in view of document

D1: Yokote Y et al., "A New Software Architecture for Evolvable Multimedia Software", Sony Corporation, Research Center.

The European Search Report cites a document listing technical papers by one of the authors of D1, Jun-ichiro Hagino/Itoh, which mentions D1 at no. 88 as having been published in the Proceedings of the European Conference on Multimedia Applications, Services and Techniques (ECMAST96), Louvain, Belgium, May 1996.

II. Notice of appeal was filed on 16 December 2008, the appeal being paid on the same day. A statement of grounds of appeal was received on 13 February 2009. The appellant requested that the decision be set aside and that a patent be granted based on claims 1-14 according to a main or an auxiliary request as filed with the grounds of appeal.

III. With a summons to oral proceedings the board informed the appellant of its preliminary opinion. The board raised an objection under Article 76 (1) EPC 1973 but otherwise indicated that the independent claims of both requests appeared to show the required inventive step over D1, Article 56 EPC 1973.

IV. In response, with letter of 3 June 2013, the appellant filed amended claims 1-14 according to both the main and the auxiliary requests. The other application documents are as follows:

description, pages

1, 4-40	as originally filed
2	filed by telefax on 29 August 2007
2a, 3	received with letter dated 18 December 2007

drawings, sheets

1/20-20/20	as originally filed

V. By telefax on 3 July 2013, the board raised an inventive step objection to which the appellant responded the next day. The board then cancelled the oral proceedings.

VI. Claim 1 of the main request reads as follows.

"A data processing system comprising a first and a second client device in communication with one another, wherein the first client device comprises:

a) storage means which stores:

an application program constructed from a plurality of application program objects;

a system object group constructed from a plurality of execution environment objects providing an execution environment that is compatible with said application program;

a first data structure (52) that specifies an application program interface for said system object group; and a second data structure (51) that shows one execution state of said application program and points to said first data structure;

a compiler operable to dynamically compile said application program into an intermediate code form or into a binary form;

b) first execution means (MVM 31a) operable to interpret and execute said intermediate code form of said application program; and

c) second execution means (MK 31b) operable to interpret and execute said binary code form of said application program and said system object group; and

wherein said first execution means (MVM 31a) and said second execution means (MK 31b) are arranged to control execution of said application program on the basis of said first data structure and said second data structure; and

the second client device comprises:

d) storage means which stores:

a second application program constructed from a plurality of application program objects;

a second system object group constructed from a plurality of execution environment objects providing an execution environment that is compatible with said second application program;

a third data structure (52) that specifies an application program interface for said second system object group; and

a fourth data structure (51) that shows one execution state of said second application program and points to said third data structure;

e) means for migrating an object located in the first client device to the second client device; and

f) compatibility checking means for checking whether an object to be migrated from said first to said second client device can be executed after migration, such that if an object can be executed after migration, the object is migrated from the first client device to the second client device, and if an object could not be executed after migration, the object is not migrated from the first client device to the second client device."

Claim 6 of the main request reads as follows:

"A data processing method comprising the steps of:

storing in a first client device an application program constructed from a plurality of application program objects;

storing a system object group constructed from a plurality of execution environment objects providing an execution environment that is compatible with said application program;

forming a first data structure (52) that specifies an application program interface for said system object group; and

forming a second data structure (51) that shows one execution state of said application program and points to said first data structure;

using a compiler operable to dynamically compile said application program into an intermediate code form or into a binary code form;

providing a first execution means (MVM 31a) operable to interpret and execute said intermediate code form of said application program; and providing a second execution means (MK 31b) operable to execute said binary code form of said application program and said system object group;

wherein said first execution means (MVM 31a) and said second execution means (MK 31b) are arranged to control execution of said application program on the basis of said first data structure and said second data structure;

the method further comprising the steps of:

constructing, in a second client device, a second application program from a plurality of application program objects;

constructing, in the second client device, a second system object group constructed from a plurality of execution environment objects providing an execution environment that is compatible with said second application program;

forming, in the second client device, a third data structure (52) that specifies an application program interface for said system object group; and

forming, in the second client device, a fourth data structure (51) that shows one execution state of said application program and points to said third data structure;

migrating an object located in the first client device to the second client device over a communication link in accordance with a compatibility check;

performing said compatibility check to determine whether said object to be migrated from said first client device to said second client device can be executed after migration, such that if said object can be executed after migration, the object is migrated from the first client device to the second client device, and if said object could not be executed after migration, the object is not migrated from the first client device to the second client device."

In view of the result of this decision the wording of the claims according to the auxiliary request is irrelevant.

Reasons for the Decision

The status of D1 as prior art

1. The present application was filed as a divisional of European patent application no. 96305139, filed on 12 July 1996 and claiming the priority of Japanese patent application no. 07-178625 filed on 14 July 1995.

1.1 The present application is considerably extended over the Japanese application, specifically by figures 12-23 and the corresponding disclosure (starting on page 31, last par., of the description as originally filed, page 31). Moreover, the present claims are based on this added matter, witness specifically the claimed first and second data structures 51 and 52 which are depicted in figures 14 and 16 and the first and second execution means MVM 31a and MK 31b which are depicted in figures 12 and 13.

1.2 The Japanese application thus does not disclose the same invention as presently claimed so that the claimed priority is not valid for the present claims (see Article 87 (1) EPC 1973) and does not have the effect provided by Article 89 EPC 1973. The effective filing date of the present application therefore is the filing date 12 July 1996 of the earlier European application.

2. D1 does not itself bear a publication date, but the European search report contains an indication of the fact that the paper was presented at a public conference in May 1996 (see point I). Based on this information, the appellant has not challenged that D1 was prior art for the present application in the sense of Article 54 (2) EPC 1973. Rather, the appellant specifically confirmed in its submission dated 3 June 2013 (p. 2, penult. par.) that D1 was the correct starting point for the assessment of inventive step. The board concurs.

Article 76 (1) EPC 1973

3. The board's objection under Article 76 (1) raised in the annex to the summons to oral proceedings was essentially based on the fact that originally filed claims 1, 6, 7 and 12 and the pending independent claims 1 and 6 specified the "second" and "fourth data structure[s] (51)" to contain "execution *conditions*" of respective application program whereas the description - both of the present and of the earlier application (see p. 34, last par. - p. 35, 1st par.) - rather disclosed the "context (51) [to show an] execution *state*". Since in the present claims the references to "execution conditions" were replaced by references to "execution state", this objection has become moot.

Article 123 (2) EPC

4. The objection under Article 123 (2) in the decision under appeal (reasons 1.1) was based on the argument that

migration of objects to a client device was disclosed but not, as then claimed, migration of an object *to an application program* in a client device. Since the claims now refer to object migration from a first to a second client device, this objection has become moot, too. The board also has no objections of its own under Article 123 (2) EPC.

The invention

5. The application generally relates to the development, deployment and modification of software in a client/ server system (see description, pp. 1-2, and fig. 1). The clients are disclosed to be preferably set-top boxes (STB).

5.1 For all computing devices in this system, server and clients alike, an object-oriented architecture is disclosed. Application programs and their execution environments consist of "objects" and an application programming interface (API) is provided between both.

5.2 Programs are compiled into intermediate code (I-code) which may either be interpreted and executed directly by what is called a Micro Virtual Machine (MVM), or which is further compiled into native code for execution by what is called the Micro Kernel (MK) (see p. 31, last par. - p. 33, penult. par.; p. 34, penult. par.; figs. 12-13). The system also comprises a layer of so-called "personality objects" need to "provide various" operating systems "OSs or virtual machines". By way of example, it is suggested that "personality objects for BASIC programming" are needed to "execut[e] intermediate code obtained by compiling [a] BASIC program" (p. 32, penult., par.).

5.3 The run-time system uses *inter alia* two data structures, one called "context" and one "descriptor". The context structure represents the execution state of a program on the MVM (p. 34, last par. - p. 35, 2nd par.) and points to the descriptor structure which in turn identifies "the API of the Personality object" (p. 35, 3rd par.; fig. 14).

5.4 The application is further concerned with the "migration" (or "shifting") of objects between devices. Typically, objects are downloaded from a server to a client, but they may also be shifted within a client (see figs. 7 and 8) or between servers, between clients, or from a client to a server (p. 39, 4th par.).

5.5 Before an object is migrated, the involved devices carry out a negotiation about whether a migration is possible and desirable (see p. 25, 3rd par.). It is disclosed that it may not be desirable to migrate an object into a "meta-object space" if the target device lacks required functionality such as certain hardware drivers or a virtual memory structure (p. 26, 1st full par.). To address this, the application therefore discloses a compatibility check to determine whether an object can be executed "even after being shifted", and that, if this is not the case, "object migration is not carried out" (p. 26, penult. par. - p. 27, 2nd par.).

Article 56 EPC 1973

6. It is common ground between the board and the appellant that D1 is a suitable starting point for the assessment of inventive step.

6.1 D1 was co-authored by the present inventor when working for the present applicant and is, in fact, closely related with the present application.

6.2 D1 also relates to a system comprising a server and several set-top boxes as clients, and discloses the same object-oriented system architecture with applications and their execution environments consisting of objects (see *e.g.* D1, fig. 2). Execution environments in D1 are called "metaspaces". Like in the application, D1 discloses a Micro Virtual Machine and a Micro Kernel MVM/MK (see *e.g.* sec. 5.3), and the data structures "context" and "descriptor"; the "context" is defined as an "execution instance for a software object" and the "descriptor" as a "data structure containing a lookup table for the methods implementing the objects in the metaspace (secs. 5.1, 5.2 and 6.1; fig. 3). Moreover, D1 discloses the idea of object "migration" either from a server to an STB (download; see sec. 8.4) or within a computing device ("intra host", see sec. 6.2, p. 11, 2nd par.).

Main Request

7. The subject matter of independent claims 1 and 6 differs from D1 in that

(i) the first and third data structures specify application programming interfaces (API) to the system object groups on the first and second client devices, respectively; that

(ii) in the first client device the second data structure points to the first data structure, as in the second client device the fourth data structure points to the third on; that

(iii) object migration is from a first client device to a second client device; and that

(iv) there is a compatibility check to determine whether an "object to be migrated ... can be executed after migration", i.e. on the second client device, and if this is not the case, the object is not migrated in the first place.

Differences (ii)-(iv) correspond essentially to differences (i)-(iii) determined in the decision under appeal in view of the then pending claims (see reasons 3.2-3.4). In the letter dated 3 June 2013 (p. 2, penult. par.) the appellant indicated agreement with this analysis and that it had "no further difference to highlight".

Re. (i) and (ii)

8. D1 discloses that a metaspace is "characterized by its descriptor" (see sec. 5.1) which contains a "lookup table for the methods implementing the objects in the metaspace" (see sec. 6.1). The board considers it to be a matter of common programming practice to provide

access to these methods through an API instead (feature i). D1 further discloses that "each context holds pointers to objects implementing the metaspace of that context" (p. 7, 1st sentence). Replacing the pointers to individual objects of a metaspace by a pointer to the descriptor of that metaspace, which itself makes the objects accessible, constitutes a simplification of the context data structure at the price of an only indirect object access via the descriptor. In the board's view, the skilled person would contemplate this simplification and the trade-off it involves as a matter of common practice and without exercising an inventive step (feature ii).

Re. (iii)

9. The board considers that the desire to transfer objects directly between clients arises naturally in the context of D1 and without in itself requiring an inventive step.

9.1 Imagine, for example, a situation in which two clients (say, two set-top boxes in the same household) are much closer to each other than either is to the remote server and in which the server connection is slow or unreliable or both. Imagine further that one the clients already runs an application which the user would want on the other client as well. Then the board would consider it an obvious desire to download this application (and its objects) directly between the clients because this would most likely be faster than via the server.

9.2 In the grounds of appeal (p. 2, last par.), the appellant argues that a "complete redesign of the system of D1 ... would be required" to enable the communication between clients without however making precise what such redesign would involve.

9.3 Also the description itself mentions the "downloading" of objects between clients only in a short paragraph (see p. 39, 5th par.) without disclosing that the claimed system would have to be substantially modified - and if so, in what manner - in order to enable the download between clients as opposed to the download from a server to a client.

9.4 The board finds this brevity appropriate on the understanding that indeed no or only obvious such modifications are needed, in view of the fact that clients and servers all provide the same object-oriented environment across which object migration is possible in any direction.

9.5 The board therefore considers that the skilled person starting from D1 would find that object migration between clients is essentially the same as between server and client since all devices implement the same environment and concludes that feature (iii) also does not establish an inventive step.

Re. (iv)

10. D1 discloses that an application existing on a specific client STB may require certain functionality to run and therefore "request the download of the appropriate metaspace and migrate to it" (see sec. 8.1, 2nd par.). In the board's understanding, this refers to a migration within a single client. D1 also discloses that, "when a

customer requests a new application, the objects implementing this application are downloaded to the customers's [client] STB" (sec. 8.4, first sentence). If "the correct metaspace already exists on the STB to run this application", the object is migrated directly "to that metaspace" (sec. 8.4, second sentence). If not, the server "accesses the feature set of current metaspaces existing on the STB and builds one from an already existing one by extending it, or by deriving a new metaspace" (see sec. 8.4, 2nd par.).

10.1 If the creation of a suitable metaspace were to fail, the objects to be migrated could not executed on the target STB. D1 however does not disclose or imply that the provision of a suitable metaspace might fail, let alone what would have to be done if it happened nonetheless.

10.2 It might be argued to be obvious for the skilled person in view of D1 to provide a means to check whether a suitable metaspace is available on the target STB and to terminate object migration to the target STB if this were not the case. This question can however be left open because the claimed "compatibility check" goes beyond a check of whether a required metaspace is available on the target STB.

10.3 According to claims 1 and 6, the "compatibility check" is for "checking whether an object to be migrated ... can be executed *after* migration" (board's emphasis). In conformance with the description (p. 26, 1st full par.) the board interprets this to mean that the checking concerns conditions at the target STB other than the existence of an environment (*i.e.* a metaspace according to D1) into which the object can be migrated.

10.4 Not migrating objects which cannot be executed anyway avoids cluttering the client's memory. The objective technical problem solved by difference (iv) thus can be said to be, as the appellant suggests (grounds of appeal, p. 3, 3rd par.), to improve the memory usage of the system, especially at the receiving client side.

10.5 In the board's view the immediate solution to this problem would be that the non-executable object be erased from the receiving client's memory.

10.6 Given further that D1 neither discloses any "compatibility check" preceding migration nor addresses whether objects can be executed "after migration", the board concludes that the claimed compatibility check is not obvious for the skilled person in view of D1.

10.7 The board therefore concludes that claims 1 and 6 of the main request are inventive over D1 by virtue of difference (iv).

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.

2. The case is remitted to the department of first instance with the order to grant a patent based on the following documents:

claims, no.	
1-14	according to the main request filed with letter dated 3 June 2013

description, pages	
1, 4-40	as originally filed
2	filed by telefax on 29 August 2007
2a, 3	received with letter dated 18 December 2007

drawings, sheets	
1/20-20/20	as originally filed

The Registrar:	The Chairman:
B. Atienza Vivancos	D. H. Rees