PCT

WORLD INTELLECTUAL PROPEI
International Bu

INTERNATIONAL APPLICATION PUBLISHED UNDER WO 9608765A1

(51) International Patent Classification 6 :
G06F 9/44

A1

(11) International Publication Number: WO 96/08765

(43) International Publication Date: 21 March 1996 (21.03.96)

(21) International Application Number: PCT/CA95/00513

(22) International Filing Date: 15 September 1995 (15.09.95)

(30) Priority Data:
08/306,481 15 September 1994 (15.09.94) US

(71) Applicant *(for all designated States except US)*: VISUAL EDGE SOFTWARE LIMITED [CA/CA]; Suite 100, 3950 Cote Vertu, Saint Laurent, Quebec H4R 1V4 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: FOODY, Daniel, M. [CA/CA]; Apartment PH 211, 625 De Maisoneuve, Montreal, Quebec H3H 2N4 (CA). FOODY, Michael, A. [CA/CA]; 21 Gabriel Roy, Nunn's Island, Quebec H3E 1M3 (CA).

(74) Agent: DUDLEY, Bruce; Gowling, Strathy & Henderson, Suite 2600, 160 Elgin Street, Ottawa, Ontario K1P 1C3 (CA).

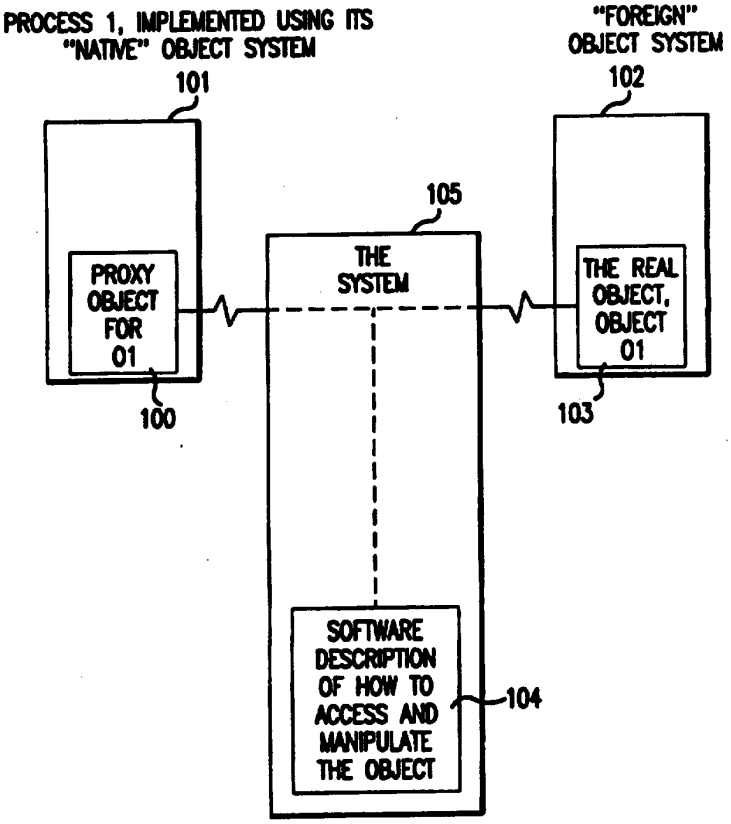
(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

Published
With international search report.

(54) Title: SYSTEM AND METHOD FOR PROVIDING INTEROPERABILITY AMONG HETEROGENEOUS OBJECT SYSTEMS

(57) Abstract

A system and method in accordance with a preferred embodiment enable objects from two or more heterogeneous object systems in a digital computer to interoperate and be combined in the creation of a larger object-oriented software project, as well as uses of such system and method. Objects from a foreign object system are unmodified, yet appear to be native to the object system in which they are used or accessed. A native proxy object (indistinguishable from other native objects) is constructed for the real foreign object. The proxy object contains an identifier to the real object, as well as a pointer to a software description of how to access and manipulate the object - e.g. how to call its methods, set its properties, and handle exceptions. When the proxy object is manipulated, it follows the instructions in the software description which, in turn, results in the corresponding manipulation of the foreign object.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

SYSTEM AND METHOD FOR PROVIDING INTEROPERABILITY AMONG HETEROGENEOUS OBJECT SYSTEMS

Technical Field

The present invention relates to object-oriented software systems and related methods for digital computers.

Background Art

Using object-oriented software techniques, software applications for digital computers are created by combining software objects. To facilitate this process, object-oriented software systems typically provide an architecture specification, called the object model, which enables all objects developed to the specification to work together seamlessly in an application. Examples of object models would include the Object Management Group's Common Object Request Broker Architecture (CORBA), and Microsoft's Common Object Model (COM.) Such systems also typically provide software, called the object system, which implements the basic features provided for in the object model.

There are numerous object systems, some very general in nature such as Microsoft's Object Linking and Embedding (OLE) (which follows the COM object model), or IBM's Distributed System Object Model (DSOM), and Iona's ORBIX, (which both follow the CORBA object model). See for example: the OLE 2 Programmers Reference, Volume 1 and 2, Microsoft Press, 1994; the IBM SOMobjects Developer Toolkit V2.0, Programmers Reference Manual, 1993; Iona ORBIX, Advanced Programmers Guide, 1994; and The Common Object Request Broker: Architecture and Specification Ch. 6., OMG, 1991; these references are hereby incorporated herein by reference.

Other object systems are designed to provide specific functionality, for example, in areas such as groupware or relational database - e.g. Lotus Notes. Still other object systems are specific to particular to applications - e.g.

Novell's AppWare Bus, Hewlett Packard's Broadcast Message Server, and Microsoft Visual Basic's VBX object mechanism. See for example: the Lotus Notes Programmers Reference Manual, 1993; the Novell Visual AppBuilder Programmers Reference Manual, 1994; the Hewlett Packard Softbench BMS, Programmers Reference Manual, 1992; and Microsoft Visual Basic 3.0 Professional Features Book 1, Control Development Guide, 1993; these references are also hereby incorporated herein by reference.

In creating a software application it is desirable to combine objects from various object systems, because different object systems are best suited to different tasks, and because the best solution is usually built from the best parts (i.e. objects.) However, objects from various object systems don't naturally work together for a number of reasons.

Object systems are rendered incompatible due to differences in the means by which objects are created, methods are called and properties are set in each object system, including differences in the fundamental mechanisms used as well differences in low-level calling conventions such as the physical layout of types and classes. For example at the fundamental level, some object systems, such as COM, use direct C++ calling mechanisms. Others such as DSOM pre-process source code so that in place of a direct call, a function from the object system is called which, in turn, returns a pointer to the real method. This pointer is dereferenced to actually call the method. Still other object systems such as OLE Automation provide specialized functions developers must use to call methods (this is often referred to as a Dynamic Invocation Interface or DII). These functions take the method to be called as an argument, as well as the method's arguments (usually packed into a particular format), and they call the method for the developer. There are numerous other broad differences and

variants in fundamental calling mechanisms. Each of these fundamental mechanisms also differ in detail. For example, CORBA requires an environment pointer argument (and has an optional context argument), while other object systems do not.

In addition to the vast differences in fundamental calling mechanisms, there are many differences in low-level calling conventions, sometimes referred to as procedure calling conventions. For example, different object systems handle the return value from methods differently when the type of the return value is a float or a structure. In one case the value may be returned on the processor stack, while in another the value may be placed in a register. Thus, using the return value of a method from a different object system would result in an error. Other examples of differences in procedure calling conventions would include how structures are packed into memory, and how arguments are placed on the stack.

Various object systems also support various types which may not be compatible with other object systems. Simple examples of types include language types such as ***integers***, ***floats***, etc. More complex language types include ***arrays***, ***strings***, and ***objects***. There are also semantic types such as "variable types" like the CORBA ***Any***, and the COM ***VARIANT***. Semantic types differ from language types in that they have a particular semantic meaning to the system. While certain semantic types may conceptually mean the same thing among various object systems, their corresponding language representation and implementation may be entirely different. A common example is strings. In COM, strings are represented using a "BSTR" (a non-NULL terminated string which contains length information), while in CORBA, strings are the traditional C language byte array (NULL terminated with no length information). As a result, a COM object couldn't pass a BSTR to a CORBA object because any functions

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.