

**MOSAIC OBJECTS AND METHOD FOR
OPTIMIZING OBJECT REPRESENTATION
PERFORMANCE IN AN OBJECT-ORIENTED
REPRESENTATION SYSTEM**

This is a continuation of application Ser. No. 07/589,114, filed Sept. 27, 1990, now abandoned.

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

BACKGROUND OF THE INVENTION

This invention relates to the field generally known as artificial intelligence and more particularly to programming in frame-based systems or more generally to object-oriented systems. Specifically, this invention relates to techniques for software development of an object-oriented system.

In the past, optimization of software-based systems and specifically of object-oriented systems has been dependent on the developer. While no solution for general software-based systems is known, characteristics of object-oriented systems have features which, according to the invention disclosed hereinafter, permit developer-independent optimization.

Many known computer languages support static data representation. Fewer languages support dynamic data representation. In static data representation, the decision on the internal layout of the data structure and representation is made at compile time of the computer program. All static data representation systems require prior specification of individual language, of each individual field and of each data type, such as integer, floating point, aggregate or the like. This greatly reduces flexibility of programming and data handling and extends the development time, although it does provide the best possible performance, since data access characteristics are all determined at compile time. An example of an object-oriented language with static data representation capability is the C++ language.

In a language capable of dynamic data representation, the decision on internal layout of data structure is deferred until run time of the program, which allows efficient applications development. The dynamic representation capability is an important distinction over static data representation. The proto-typical example of an object-oriented language capable of dynamic data representation is the SmallTalk language of Xerox Corporation. Because of frequent and extensive lookup at program run time, the typical system with a dynamic data structure yields significantly slower run-time performance than a comparable system having a static data structure. Although it allows for easy applications development, SmallTalk does not run many practical applications in a generally acceptable amount of time. As a consequence, programmers may use SmallTalk or other similar languages with dynamic data structures during system development but thereafter recode a production system in another language which supports static data representation.

It is therefore desirable to provide a programming environment which is easy and flexible for development of a software-based system and which does not require extensive reengineering in a different language for production of a software-based system.

The problem area herein addressed is not to be confused with the area of database management, where database management systems employ dynamic look-up capabilities as part of an indexing scheme on a revisable database. The problem addressed herein is how to minimize recoding of source code in a generalized object-oriented programming environment.

What is needed is a programming environment and methodology that allows delay until compile time of any specification about how static the structure is to be, and what is needed is a range of options between the formation of a dynamic data structure and the formation of a fully static data with commensurate improvements over a program with fully dynamic data representation.

Definitions

In order to more fully understand the invention, an understanding of certain terms is helpful. These definitions are not necessarily all-inclusive, and it should be understood that even within certain audiences of the relevant arts, common concepts employ different terminology and common terms have various meanings.

An "object" is a specific term of art, sometimes referred to as a "unit" in the literature, which is a data structure. It is analogous to a "row" in a relational database. Typically the term "object" refers to data structures which carry both attributes ("slots") and behaviors or methods.

An "attribute" is a specific term of art, sometimes referred to as a "slot" in the literature, which is a "property" of an "object," such as a method associated with an "object." A parallel concept is the "fields" of a data structure. The artificial intelligence community refers to the concept by both terms.

A "frame" is an "object" with some additional capabilities, such as secondary structure, or attributes of attributes, active capabilities called for example "monitors," "active values," or "triggers," and self-descriptive capabilities, such as an understanding what attributes it has, what its parents are or what its parents or children are.

A "frame-based system" is a system of frames, or in other words, a system which has a self-descriptive capability.

An "object system" is a "frame-based system" which in many cases does not have the ability to describe itself.

A "value" is a term of art defining what is "in" an attribute ("slot") at any particular moment.

A "selector" is a reference to an "attribute." It is a name given to an "attribute" and the way by which access to the "attribute" is specified.

Other definitions have been or will be introduced as is helpful for an understanding of the invention.

SUMMARY OF THE INVENTION

According to the invention, an object-oriented software-based system is optimized by providing a data representation which is initially permitted to be completely dynamic, that is, a decision on layout of internal data structure is deferred such that at compile time during development data need not be specified and thereafter the data structure and representation is progressively specified and optimized. The invention is