Journal of International Information Management

Volume 3 | Issue 2 Article 6

1994

Smalltalk: Big things forecast

Brian D. Lynch
University of Wisconsin- La Crosse

Smalltalk: Big things forecast

Cover Page Footnote

The fundamental objective of this paper is to provide an overview of Smalltalk's recent emergence as a major programming language. Smalltalk's growth in popularity is being driven by information systems professionals' increasing interest in learning and applying object-oriented technology. The paper proceeds as follows: first, the historical significance and commercialization of Smalltalk is discussed; next, an overview of the object-oriented paradigm is presented; then, pure and hybrid object-oriented languages are contrasted; and finally, a brief summary of Smalltalk's history and future is presented.

Smalltalk: Big things forecast

Brian D. Lynch
University of Wisconsin-La Crosse

ABSTRACT

The fundamental objective of this paper is to provide an overview of Smalltalk's recent emergence as a major programming language. Smalltalk's growth in popularity is being driven by information systems professionals' increasing interest in learning and applying object-oriented technology.

The paper proceeds as follows: first, the historical significance and commercialization of Smalltalk is discussed; next, an overview of the object-oriented paradigm is presented; then, pure and hybrid object-oriented languages are contrasted; and finally, a brief summary of Smalltalk's history and future is presented.

THE HISTORICAL SIGNIFICANCE OF COMMERCIALIZATION OF SMALLTALK

The first language developed purely to support object-oriented programming was Smalltalk (Martin, 1993). It was during the development of Smalltalk that the term object-oriented originated (Goldberg & Robson, 1983). However, the conceptual foundations for object-oriented technology dates back to the early 1960s and the development of the Simula programming language (Martin, 1993).

Guided by the central ideas of Simula, Alan Kay while a graduate student at the University of Utah in the late 1960s did research related to object-oriented technology. Then in the early 1970s, Kay went to Xerox's Palo Alto Research Center (PARC) and developed Smalltalk (Martin, 1993). Adele Goldberg and Daniel H. H. Ingalls are also credited with having made key contributions to Smalltalk. Over time, Smalltalk evolved into Smalltalk-76 and then Smalltalk-80 with versions developed by Xerox and others to run on a number of different computers (Meyer, 1988). In the mid 1980s, Goldberg championed the formation of ParcPlace Systems as an independent company to make Smalltalk a more commercially viable product (Garber, 1993). Today, the two leading Smalltalk vendors are Digitalk and ParcPlace Systems (Verity, 1993).

From a historical perspective, Smalltalk's development played a key role in the emergence of graphical user-interfaces (GUIs)--composed of graphical objects such as overlapping windows, icons, pop-up and pop-down menus, and buttons to be clicked with a mouse. For example, Steven Jobs' viewing of Smalltalk during a 1979 visit to Xerox PARC provided him with the idea that eventually led to the Apple Macintosh and its ground-breaking GUI approach (Verity, 1993).

Today, a convergence of technological advances—such as the growth in the use of GUIs and more powerful desktops—has furthered the growth and penetration of object-oriented languages. GUIs like Apple's Macintosh and Microsoft's Windows lend themselves to the utilization of the object-oriented approach for systems development (The, 1992).

Currently, the dominant object-oriented development environment is C++. The C++ market is roughly 10 times the size of the Smalltalk market (Bozman, 1993). But Smalltalk is projected to have a 30% market share by 1997 (Radding, 1994). A survey of 291 corporate systems developers object-oriented plans by Market Perspectives provides additional evidence of Smalltalk's commercial emergence. Twenty-eight percent planned to be involved with Smalltalk in the next two years (Ballou, 1994). Further, IBM plans to implement the Smalltalk language across all its hardware and operating systems platforms by the end of 1995 (Stedman, 1994).

AN OVERVIEW OF THE OBJECT-ORIENTED PARADIGM

Complete agreement among information systems professionals as to what exactly constitutes object-oriented software is still lacking (McGregor & Sykes, 1992). Nonetheless, five concepts underlie most discussions of the object oriented paradigm: objects, classes, inheritance, encapsulation and polymorphism (e.g., Davis, 1994; Jacobson et al., 1992; Martin, 1993; McGregor & Sykes, 1992; Melymuka, 1994; Parker, 1993; Wilde & Huitt, 1992).

An object represents a single instance of a person, a place, or a thing. Objects have data (attributes) and methods (processes) associated with them. Similar objects are grouped together to form classes or object types—e.g., beagle and collie could be considered objects in a class known as dog. Similar classes can be grouped together based on their common characteristics to form class hierarchies. The key is to organize class hierarchies following a generalized to specialized continuum—e.g., mammal->dog->beagle.

Inheritance permits a class lower in a class hierarchy to inherit the attributes and methods of classes above it in the class hierarchy. Thus, when defining a new class in a class hierarchy, only the 'new' characteristics need be defined—i.e., those attributes and methods not present in the classes above it in the class hierarchy.

Encapsulation or data hiding requires that an object's data and implementation details are hidden from other objects. Thus, an object's data can only be accessed through its own methods. Encapsulation prevents an object's data from being corrupted or misused, because it blocks users from accessing the data except through the object's own methods.

Literally, polymorphism means many or multiple forms. With respect to the object-oriented paradigm, polymorphism deals with messages or communications (e.g., a request for information or an action) between objects. Polymorphism permits the same message to be understood by different classes. Further, the response (method) implementation does not have to be the same across the different classes—i.e., different classes may respond in different ways to

the same message. Polymorphism implies that the receiver of a message (as opposed to the sender) determines how a message shall be interpreted. For example, different classes of three dimensional geometric figures would have different method implementations for responding to message requests for surface area or volume.

PURE VERSUS HYBRID OBJECT-ORIENTED LANGUAGES

Several traditional languages (based on the top-down functional design approach to systems development) have had object-oriented capabilities added to them—e.g., C++ which represents an extension of C. C++ and languages like it are referred to as hybrid languages. Hybrid languages can use their traditional compiler by first using a preprocessor: object-oriented language->preprocessor->traditional language->compiler->machine code. An advantage of using a hybrid language is that programmers can learn an extension of something they already know as opposed to having to learn a complete new language. However, given that hybrid languages support traditional structured thinking as well as object-oriented thinking, many programmers end up using hybrid languages in traditional ways and do not make the shift to object-oriented thinking and programming (Martin, 1993).

Ordinary programmers also find C++ difficult to learn, taking up to 18 months to become proficient. Whereas, programmers have far less difficulty in mastering Smalltalk. Programmers can become productive within weeks and proficient in six months or less (Garber, 1993). When used properly, pure object-oriented languages like Smalltalk have many advantages over traditional programming languages and approaches: Easier learning, reduction of complexity (a McCabe metric of 3 as opposed to 10), easier debugging, reusable classes, reusability due to inheritance, complexities hidden by encapsulation, easier to make changes, and thus, greater creativity (Martin, 1993).

Historically, pure object-oriented languages have given lower machine performance than traditional or hybrid languages. However, better design of optimizing compilers for pure object-oriented languages is significantly reducing this advantage (Martin, 1993).

SUMMARY

Given that Smalltalk was the early trailblazer in the object-oriented area, it is only fitting that Smalltalk should be reemerging as a key language now that the object-oriented paradigm is growing in popularity. Commercially, Smalltalk and its users are now poised to benefit from the conceptual seeds that Smalltalk first sowed over two decades ago. In conclusion, the growing

utilization of the object-oriented paradigm by information systems professionals coupled with the aggressive marketing of Smalltalk platforms by firms such as Digitalk, ParcPlace and old information technology stalwarts like IBM point to a bright future for Smalltalk and those versed in it.

REFERENCES

Ballou, M. (1994, March 28). Vendor promises puzzle users. *Computerworld, 28*(13), 61-62.

Bozman, J. S. (1993, November 29). Pact excites Smalltalk users. *Computerworld, 27*(48), 75-76.

Davis, W. S. (1994). *Business systems analysis and design*. Belmont, CA: Wadsworth Publishing.

Garber, J. R. (1993, April 12). Working faster. *Forbes*, 110.

Goldberg, A. & Robson, D. (1983). *Smalltalk-80: The language and its implementation.* Reading, MA: Addison-Wesley.

Jacobson, I., Christerson, M., Jonsson, P. & Overgaard, G. (1993). *Object-oriented software engineering: A use case driven approach.* Workingham, England: ACM Press of Addison-Wesley.

Martin, J. (1993). *Principles of object-oriented analysis and design.* Englewood Cliffs, NJ: P T R Prentice Hall.

McGregor, J. D. & Sykes, D.A. (1992). *Object-oriented software development: Engineering software for reuse.* New York, NY: Van Nostrand Reinhold.

Melymuka, K. (1994, March 21). Getting to "aha!" *Computerworld, 28*(12), 99-108.

Meyer, B. (1988). *Object-oriented software construction.* London, Great Britain: Prentice Hall International.

Parker, J. (1993, March 15). Windows developers get object lessons. *Datamation, 39*(6), 94-97.

Radding, A. (1994, May 30). Smalltalk sizzle. *Computerworld, 28*(22), 103.

Stedman, C. (1994, May 16). IBM bets big on Smalltalk. *Computerworld, 28*(20), 16.

The, L. (1992, November 1). Windows drives OOP on the desktop. *Datamation, 38*(22), 50-53.

Verity, J. W. (1993, April 19). Finally the buzz is about Smalltalk. *Business Week,* 111-112.

Wilde, N. & Huitt, R. (1992, December). Maintenance support for object-oriented programs. *IEEE Transactions on Software Engineering, 18*(12), 1038-1044.