8-16-1996

THE FUTURE OF APPLICATION DEVELOPMENT

Mark Hensel
University of Texas at Arlington, hensel@uta.edu

THE FUTURE OF APPLICATION DEVELOPMENT

Mark Hensel, Ph. D., CCP
Director of Information and Instructional Resources
College of Business Administration
University of Texas at Arlington
Box 19377
Arlington, TX 76019
Telephone: (817) 2723380
Fax: (817) 7945801
E-mail: hensel@uta.edu

The history of applications development has been characterized as two great cycles or 'waves' of events.

The first cycle was from 1960 to 1980 and consisted of three parts or phases. The first phase from 1960 to 1970 was a period of problems, frustrations, and failures caused by an inability to write high quality applications that met user and management needs and expectations; and by a failure to do it in a timely manner, and at budgeted cost. The crisis phase was followed by a period of proposed solutions from 1970 to 1975, and that phase was followed by a period of assimilation of the proposed solutions from 1975 to 1980 (Martin and McClure 1985).

The second cycle follows the same form but is shorter than the 20-year cycle of the first wave. The second period of problems and frustrations was from 1980 to 1985. The second crisis was caused by applications development backlogs which delayed development efforts, and made users unhappy (Martin and McClure 1985). Using the format of the Martin and McClure model, the period of proposed solutions for this second cycle was between 1985 and 1990, and the period of assimilation was between 1990 and the present.

Soon after computers were introduced into businesses some problems were caused by unexpectedly high demands for information services. The demands for computing support caused development backlogs, and many new systems failed to deliver promised solutions, to work as advertised, or to perform as expected (Sanders 1970, De Marco 1978, and Hoplin 1993).

The need for a method to develop systems became apparent after the first computing systems were introduced in organizations (Sanders 1970 and Couger 1973). That was followed by a need to have some sort of standardized, documented method of systems development so that system developers could understand what system users wanted and, hopefully, build the systems they wanted. The First Generation of Development Methods was built upon the tools and techniques available at the time (around 1950 to 1960). Document flowchart methods became process flowcharts

(Chapin 1971), a course in systems analysis was started at MIT in 1950, and machine

languages were developed into high level languages to support faster application development (Couger 1973).

The Second Generation of Development Methods began in 1960 and lasted until about 1970 (Couger 1973 and Marco and Buxton 1987). This period was characterized by the use of highlevel procedural languages and significant advances in hardware and software, and computing became a discipline in its own right (Marco and Buxton 1987). But the 1960s was also a period of frustration, problems and failures (Martin and McClure 1985). Applications development lagged behind other computing developments and it was still hard to build the right system for the right reasons and get it done on time and under budget (Marco and Buxton 1987).

The poor results of the 1960s did not end the demand for larger, faster and better systems; and hardware improvements made possible such improved systems. The 'super' computers of the Third Hardware Generation (1964 to 1975) were so much better than their predecessors that organizations began to demand more of their computer systems (Sanders 1970). It was this demand for bigger, better, faster systems which led to the industry response in the form of the management information system (MIS) concept of the 1960s, a concept which was ahead of its time. We were unable to deliver any significant MIS in the 1960s, and the MIS concept was the 'flop' of the 1960s (Hershman 1968) which led to many of the development revisions of the 1970s (Ackoff 1971).

The Third Generation of Development Methods started in 1970 and lasted until 1980 (Couger 1973 and Marco and Buxton 1987). This period was characterized by the advent of the microprocessor and by attempts to write better, higher quality software (Marco and Buxton 1987) as solutions were sought to cure the development problems of the 1960s (Martin and McClure 1985).

The Fourth Generation of Development Methods began in 1980 and has lasted to the present. The fourth generation is characterized by a movement to more development tools and techniques to augment the methodologies of the third generation, to speed up the development of applications, and to improve the quality of applications.

The fourth generation is marked by a new period of problems and frustrations from 1980 to 1985 which were caused by an increasing backlog of applications waiting to be developed (Martin and McClure 1985). That period was followed by a period of proposed solutions from 1985 to 1990, and that period was followed by a period of assimilation from 1990 to the present time.

We may be entering a third wave or cycle of problems, proposed solutions, and assimilation. It may be that this repeating cycle of frustration, development, and assimilation will continue forever because we have never been satisfied with applications development. The tools, techniques and methodologies that were supposed to cure the backlog problems identified in the early 1980s have had limited success. In spite of the many improvements in applications development methods, applications development failures are still associated with the traditional problems: Lack of user and managerial involvement, unclear requirements for new systems, slow applications development, and poor implementation (Modha and Bruce 1990).

There is growing dissatisfaction with many of the commercial software products available at this time. Many of those commercial applications have added more features than many people will ever use. They are called 'Bloatware' to show how they have become too big and too complex for many users to understand, and too hard to use (Cortese et al. 1995, Pree and Pomberger 1995).

Another new family of development tools, techniques, and methods will follow this period of unrest. A shift to the Object-Oriented paradigm is already under way (Lewis 1994, 1995a, 1996; LaPlante 1995; and McLendon 1996). The fascination with the Java language and its imitators promises another rush into new tools for rapid applications development (Cortese et al. 1995, Elmer-Dewitt 1996, Flynn and Clarke 1996). These and other new tools, techniques and methods will promise to solve our continuing development problems just as structured techniques promised to solve them in the 1970s; and just as 4GLs, CASE tools, and prototyping techniques promised to solve the same problems in the 1980s.

The frustration and discontent may herald the appearance of the Fifth Generation of Development Methods. As a result of the dissatisfaction with traditional development methods, three trends are emerging that could have major implications for the future of applications development.

First, some types of software are becoming disposable (Gillin 1995, Lewis 1996). That is happening because organizations demand faster development of their applications. The implication is that many organizations are willing to sacrifice quality for speed in software development because the applications developed today become obsolete in a very short time (Gillin 1995, McLendon 1996, Lewis 1996).

At the same time there is a demand for very high quality software in certain industries such as the health and medical industries, the space program, and the financial industries (Joch 1995). Those industries tend to be very stable with little change or very slow change, or they support human life or finances and do not tolerate errors. These types of organizations and industries continue to demand well designed, relatively error-free, high quality software; and they will continue to demand it (Parnas 1996).

Second, rapid applications development (RAD), object-oriented development, and network-centric computing seem to be emerging as the development methods of choice in many organizations (Martin 1991, Lewis 1995a, McLendon 1996, Flynn and Clarke 1996). The implication is that although structure is still needed to properly analyze and specify a development project (Joch 1995, Parnas 1996), speed of development is essential.

Object-oriented development may be a temporary measure because there is a software revolution under way that may be changing the way many applications are developed, packaged, and distributed. This revolution is a direct assault on the WINTELL dominance of the desktop environment and is led by a variety of small companies on the Internet and the Web (Cortese et al. 1995, Elmer-Dewitt 1996, and Lewis 1996).

Third, languages such as Sun's "Java", Bell Labs' "Inferno", and others may make it possible to gather "applets" of code from sites on the Internet and use them in new, rapidly developed applications. Browsers could become the "Windows" environment of the Web (Cortese et al. 1995). The implications are that application development may become speedier, and that the training for a career in the development field may change dramatically.

All of these trends mean that the field, the discipline of applications development is changing again, and changing very quickly. The IS industry, IS educational institutions, and IS professionals must change just as quickly to meet the demands of the future (Dickson 1996, McLendon 1996).

References and Bibliography

Ackoff, R. L. 'Management Misinformation Systems," Management Science (The Application Series), volume 14, number 4, December 1967, pp. B147 to B156.

Chapin, N. Flowcharts, Auerbach Publishing, Princeton, NJ 1971.

Cortese, A., J. Verity, K. Rebello, and R. Hof. "The Software Revolution," Business Week, December 4, 1995, pp. 78 to 90.

Couger, J. D. "Evolution of Business System Analysis Techniques," ACM Computing Surveys 1973.

De Marco, T. Structured Analysis and System Specification, Yourdon Press (PrenticeHall), Englewood Cliffs, NJ 1978.

Dickson, A. (Frito-Lay Company). "Prepare For Your Next Job In Your Current Job," panelist remarks at the DPMA National Collegiate Conference at Corpus Christi, TX on April 19, 1996.

Elmer-Dewitt, P. "Why Java Is Hot," Time Magazine, January 22, 1996, pp. 58 to 60.

Flynn, J. And B. Clarke. "How Java makes network-centric computing real," Datamation, March 1, 1996, pp. 42 to 44.

Gillin, P. (Executive editor of COMPUTERWORLD). "Strange New World," address at the ISECON '95 convention at Charlotte, NC, November 1995.

Hershman, A. 'A Mess in MIS," Dunn's Review, January 1968, pp. 26 to 27.

Hoplin, H. "Integrating Information Systems Technology in the Organization of the Future," Proceedings of the Tenth Information Systems Education Conference published by the Education Foundation of the DPMA, Park Ridge, IL, 1993.

Joch, A. "How Software Doesn't Work," BYTE Magazine, December 1995, pp. 48 to 60.

LaPlante, P. "A Retrospective Look Forward," Computer Magazine, August 1995, pp. 26 to 27.

Lewis, Ted (1994). "Where Is Computing Heading?", Computer Magazine, August 1994, pp. 59 to 63.

Lewis, Ted (1995a). "Where Is Client/Server Software Headed?", Computer Magazine, April 1995, pp. 49 to 55.

Lewis, Ted (editor, 1995b). "Where Is Software Headed? A Virtual Roundtable", Computer Magazine, August 1995, a series of articles speculating on the future of software and software development, pp. 20 to 29.

Lewis, Ted (1996). "The Next $10{,}000_2$ Years: Part II," Computer Magazine, May 1996, pp. 78 to 86.

Marco, A. and J. Buxton. The Craft of Software Engineering, AddisonWesley Publishing, Wokingham, England 1987.

Martin, J. Rapid Application Development, MacMillan, New York 1991.

Martin, J. and C. McClure. Diagramming Techniques for Analysts and Programmers, PrenticeHall, Englewood Cliffs, NJ 1985.

McLendon, J. R. (President of Texas Instruments Software Business). "Reengineering Information Technology For Change Faster Than Change," presentation to the DPMA National Collegiate Conference at Corpus Christi, TX on April 19, 1996.

Modha, J., A. Gwinnett and M. Bruce. "A Review of Information Systems Development Methodology (ISDM) Selection Techniques," OMEGA International Journal of Management Science, volume 18, number 5, 1990, pp. 473 to 490.

Parnas, D. L. "Why Software Jewels Are Rare," Computer Magazine, February 1996, pp. 57 to 60.

Pree, W. And G. Pomberger. "The Past As Prologue," Computer Magazine, August 1995, pp. 27 to 28.

Sanders, D. H. Computers and Management, McGrawHill Publishing Company, New York 1970.