

FACTORS TO BE TAKEN INTO ACCOUNT IN THE EVALUATION OF THE IMPACT OF OBJECT-ORIENTED TECHNOLOGY IN MIS DEPARTMENTS

Error! Marcador no definido.

M. Perez

Simon Bolivar University
Dept. of Processes and Systems
Caracas - Venezuela
e-mail: movalles usb.ve

ABSTRACT

The Object Oriented Paradigm (O-O), coupled with a corporate transition strategy may be the key to improving productivity and the quality of the software that is so very much sought after by MIS departments. In view of the fact that the inclusion of this new technology implies new scientific and engineering principles and problem-solving techniques, it has become necessary to start up a discussion on the factors to be taken into account in corporate technological transition strategies. Factors such as human resources, investment, methodologies, software development processes, measures, tools and organizational structures are analyzed.

KEYWORDS: Object orientation, reuse, corporate strategy, library, methodology.

1. INTRODUCTION

The incorporation of Object-Oriented (O-O) technology in the new developments in MIS departments is one of the current challenges at hand for management. A radical change is implicated in this incorporation process. Radical changes must be carried out within the framework of an adequate corporate strategy. This paper puts forward an analysis of a set of factors that should be taken into consideration at the time of formulating this strategy, factors that are strongly conditioned by underlying concepts in the object-oriented paradigm and by one of its more outstanding qualities: its potentiality for reuse. Reuse is understood to be the use of engineering knowledge based on existing systems in order to build new ones. Reuse is considered to be a technique that improves the quality of software. The factors are more non-technical than they are technical. In particular, the three most critical elements leading to a successful reuse program are management leadership and support, organizational change and the creation of a reuse mindset. These, plus a small, high-quality library, are the things to invest in. (Griss and Wosser, 1995)

2. TECHNOLOGICAL RADICAL CHANGE

When an organization decides to adopt a technological innovation, one of the most important aspects that has to be evaluated is what place will the innovation have? Is it to be an incremental change or a radical change? Incremental changes produce few modifications to the process and/or product. On the contrary, they supposedly reinforce competence. Radical changes, on the other hand, are based on new scientific or engineering principles that therefore require different technical and problem-solving skills.

MIS departments do not escape this type of evaluation, and must currently face a new challenge: the adoption of the Object-Oriented (O-O) paradigm in their developments. The situation known as "software crisis" claims a solution that could truly improve the productivity and quality of MIS. Nobody doubts that changing one single element such as a tool, a technique, a programming language or a paradigm, would not produce the desired improvement in productivity and quality. However, a perceptible use of complementary techniques and tools, coupled with a paradigm with proved advantages, may generate the expected results (Tkach and Puttick, 1994).

The solid software principles of object-oriented analysis and design, together with improvements in the languages, tools and training and the establishment of a reuse corporate strategy, are in the possibility of materializing the desired growth in productivity and required cost reduction, while at the same time improving the credibility of the projects and plans of MIS departments (Tkach and Puttick, 1994). The implementation of a radical change implies many elements of risk due to the broad range of impact that is generally unknown. Moreover, the implementation of a new and radical technology requires different strategies to manage the risk and overcome hindrances in its implementation. By adopting this new technology (the O-O), investments in conventional technology may be lost, personnel has to be re-trained and the software development process will have to be reconverted (Fichman and Kemever, 1994).

One of the more relevant qualities of this new technology is its potential for reuse. For this reason, it is considered to make some comments in this regard. The advantages of reuse are obvious, especially when compared to the developments that start from scratch. There are organizational problems and psychological reasons such as the "not invented here" (NIH) syndrome, which hinder the development of reuse. The NIH syndrome, for instance, expresses the fact that programmers tend to reconstruct rather than reuse software components. This is true if the components do not comply with the requirements (Pree, 1995). The technical aspects are seen by some authors as crucial preconditions for the improvement of reusability. Meyer for instance (Meyer, 1988) states that without an adequate technology, organizational changes are hardly useful. Routine-oriented programming languages enable the construction of libraries of reusable routines; O-O programming languages in turn, support the programming by differences and objects/classes can be adapted without affecting the source code of the original object/class. Inheritance and dynamic binding concepts are sufficient to build frameworks, which are reusable semi-finished architectures for different domains of application (Pree, 1995). These frameworks seem to be a true development in reusability inasmuch as they are not just simple code blocks, but rather an entire design subsystem that can even be reused.

One of the hottest topics in the O-O circles these days is the notion of design patterns. O-O literature is invaded with articles to this effect and the first conference on patterns has already taken place (Lalonde and Pugh, 1995). Design patterns attempt to describe the frameworks at a higher level of abstraction than that of the code that implements them. Thus, design patterns can be seen as a complement to O-O languages. Design patterns support re(use) and the development of frameworks.

3. FACTORS TO BEAR IN MIND

A discussion on the impact of this new paradigm in the MIS departments is crucial in order to formulate a transitional corporate strategy. The analysis of some of the factors to be taken into account is therefore proposed as follows, once the incorporation of O-O technology in new developments is to be the goal:

3.1 Human Resources

When O-O technology is being considered for adoption into a project, new roles have to be introduced. At a glance, the following are noticeable: The Developer: when working with this technology, evolutionary and/or incremental methodologies have to be followed and this implies getting involved in the analysis as well as the design phases. Developers are divided into those that produce reusable components and those that combine these components to build applications (also called class producers and class consumers). The Objects Modeller: The role of the objects modeller has a broader scope of influence than the equivalent role in a traditional technology. This is mostly due to the fact that in developments with O-O technology, the difference among analysis, design and implementation is blurred. This role therefore covers all the phases in the project (Tkach and Puttick, 1994). The Person responsible for the Library: in the various stages of the project, new classes may spring up or old ones existing in the library may be refined. Producer and consumer groups work closely together and therefore there is a great need to count on a system that manages the configurations when working with large, complex systems. The person responsible for the Library should be the leader of the system. Reuse Architect: shall be the leader of the development and maintenance of frameworks. He must know what services to offer to the projects and must carry out follow-up activities.

On the other hand, managers have developed skills in order to manage those human resources. The efficient management of O-O projects requires managers to define and implement a model of the reuse process (Goldberg and Rubin, 1992) to

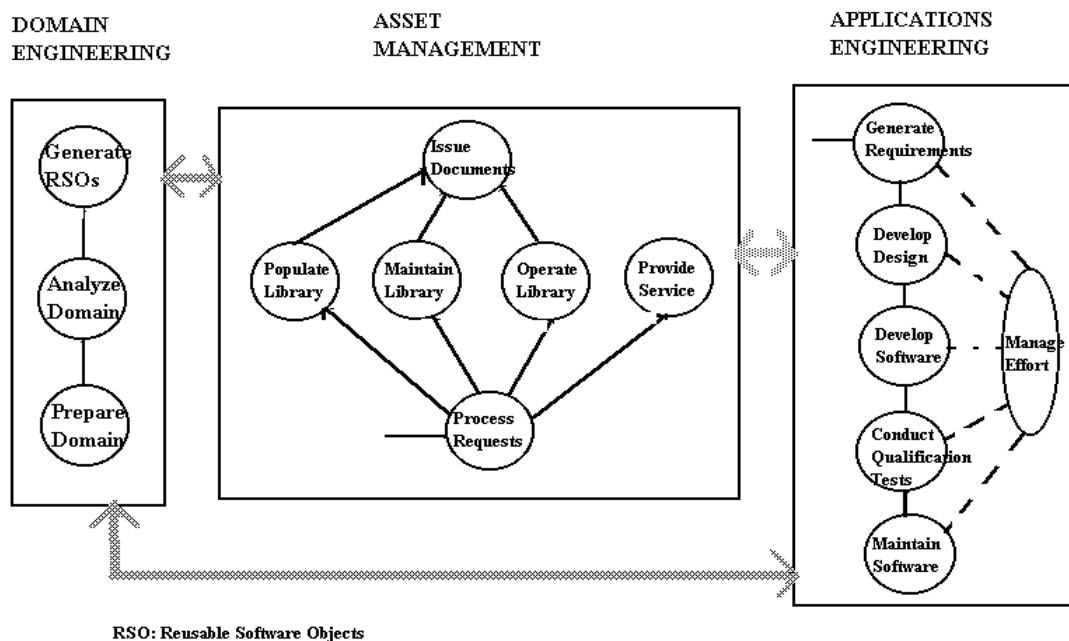
describe activities such as: deciding whether to build or to purchase a component, mechanisms to certify the quality of a component, the storage of a component, the description of its access and the dissemination of its availability.

3.2 Investment

At present, when an MIS manager acquires a third- or fourth- generation language, the general rule is that he does not give consideration to purchasing components. The usual practice is to think that the language is self-sufficient. Investment is made on tools that would increase the programmers' efficiency (editors, debuggers, etc.). In this new technology, it is necessary to think of the frameworks. For the time being, it is better to purchase the technical frameworks rather than to develop them, in light of their high quality. A good framework is fundamental and those that are going to be developed should be developed beforehand by the more experienced programmers. On the other hand, the tools for storage and search of reusable components are highly sophisticated. This leads us to conclude that the incorporation of O-O technology goes hand in hand with a heavy investment, especially in non-common aspects.

3.3 Software development processes with reuse

Reifer (Reifer, 1993) suggests the life cycle of three parallel paradigms. He proposes to undertake the Domain Engineering and Asset Management processes parallel to the traditional software development process (Applications Engineering) (see figure). Domain Engineering is the process of identifying, designing and developing reusable objects based on a knowledge of the candidate objects with a high availability of the reusable object consumer groups in an efficient and effective manner. Asset Management is the process of identifying, categorizing, cataloging, and making proven software objects available to potential users in a timely and efficient manner. Applications software Engineering is the process of analyzing, specifying, designing, implementing, integrating, testing and maintaining software the basis of which is the existence of reuse-based methodologies. Also, promotion and communication mechanisms must be established in order for the consumers of objects/classes to be continuously informed about reusable components, their meaning and purposes; and finally, prizes should be awarded for re(using) them.



3.4 Methodologies

Currently there is a wide array of methodologies being offered and there are more common factors than differences between them (Booch, 1994), (Wirfs-Brock et al, 1990), (Rumbaugh et al, 1991), (Coleman et al, 1994), (Jacobson et al, 1992), (Perez and Antonetit, 1994). The ideal situation would be for an organization to standardize its methodology. Reality shows process (Goldberg and Rubin, 1992), however, that many organizations develop their own in-house methodology and choose a certain notation, whose semantics may or may not coincide with its author's original intention. Now therefore, with the O-O technology, the prototype continues to be a tool of basic importance to confirm that the computerized model of the world that was obtained by means of interactions with the user, corresponds to the model of the world from the user's point of view.

3.5 Metrics

According to Tkach (Tkach and Puttick, 1994), it is not trivial to measure the productivity of someone who is not producing but reusing. Few are the metrics experiences and proposals in O-O technology. However, (Reifer, 1993) it sounds logical that the measuring units are not code lines but rather classes; and that the application of techniques as function points makes no sense.

3.6 Tools

As analyzed in the previous points, the incorporation of O-O technology in the development of applications must be backed by a library enabling the storage of, and access to reusable components. These libraries are highly sophisticated, have no standards and require human resources trained in their use and maintenance. This library is stated by many authors as a critical factor in the success of any organization strategy involving reuse (Frakes, 1994).

3.7 Organizational Structure

Recent results of reuse programs show that the organizational factors greatly affect the putting into practice of reuse programs (Fafchamps, 1994). A crucial organization factor in the implementation of reuse is the relationship between class producers and consumers. Different structures can be implemented for these two types of roles. Four of them will be mentioned as follows, with their advantages and disadvantages: Sole Producer: One sole producer provides reuse services to at least two teams of consumers. The disadvantage is that sole producers have no authority to arbitrate conflicts. Nested Producer: two or more of each consumer team has a member dedicated to reuse services. Its disadvantage is the dispersion of resources, although in the producer there is a space in each consumer team. Pool Producer: In this structure, two or more teams collaborate in the production and sharing of components. The disadvantage is the communication is at a high and intense level, especially in the first stages of collaboration. Highly recommendable in limited-scope reuse programs. Team Producer: This structure fits in nicely in the organization structure of divisions. The producer team has the same organizational space as the consumer teams. The disadvantage is that the producers may aspire to perfect solutions whereas the consumers only hope to have their requirements met.

4. CONCLUSIONS

The incorporation of O-O technology in the development of new applications involves a radical change. This change should be introduced based on a corporate strategy. Some of the factors that should be taken into account in the design of this strategy are: new roles and therefore personnel re-training, strong investments in tools and non-common elements, the redesign of the software development process, the acquisition of a standard methodology within the organization, metrics based on new concepts, the handling of sophisticated tools (libraries) and the adaptation of the organizational structure in line with this new technological change.

REFERENCES

Booch, G.(1994); Object-Oriented Analysis and Design with Applications; Second Edition, The Benjamin/Cumming Publishing Company, 1994.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H , Hayes, F. and Jeremaes, P.(1994); Fusion; Addison Wesley, 1994.

Fafchamps, D. (1994); Organizational Factors and Reuse,; IEEE Software, September, 1994.

Fichman, R. and Kemever, C. (1994); Object-Oriented and Conventional Analysis and Design Methodologies;IEEE October, 1992.

Frakes, W. (1994); Success Factors of Systematic Reuse; IEEE Software, September, 1994.

Goldberg, A. and Rubin, K. (1992); Object-Oriented Project Management; Tutorial; OOPSLA 92

Griss, M.; and Wosser, M. (1995); Making Reuse Work at Hewlett-Packard; IEEE Software, January 1995.

Jacobson, I., Christerson, M., Jonsson, P. and Övergaard G. (1992); Object-Oriented Software Engineering. A Use Case Driven Approach.; Adison Wesley, 1992

Lalonde, W. and Pugh, J. (1995); Communicating reusable design via Design Patterns; Journal of Object-Oriented Programming; January, 1995.

Meyer B. (1988); Object-Oriented Software Contruction; Englewood Cliffs NJ: Prentice Hall, 1988.

Perez M. and Antoneti J.(1994); _Metodologias Orientadas a Objeto: Métodos, Herramientas y Técnicas. Jornadas Informática, Tecnología y Sociedad, UCV, Venezuela; November 1994.

Pree W.(1995); Design Patterns for Object-Oriented Software Development; Addision Wesley, 1995.

Reifer , D. (1993); Software Reuse for TQM; Total Quality Management for Software. VNR Computer Library; 1993.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.(1991); Object-Oriented Modeling and Design; Prentice Hall International, 1991.

Tkach, D. and Puttick, R.(1994); Object Techonology in Application Development; IBM International Technical Support Organization; The Benjamin/Cumming Publishing Company, 1994.

Wirfs-Brock, R., Wilkerson, B. and Wiener, L.(1990); Designing Object-Oriented Software; Prentice Hall Englewood Cliffs NJ, 1990.