2005

The Benefits of Object Oriented Development: Toward a Framework for Evaluation

John W. Satzinger
Missouri State University, jws086f@smsu.edu

Duane R. Moses
Missouri State University, drm167f@smsu.edu

The Benefits of Object-Oriented Development: Toward a Framework for Evaluation

John W. Satzinger
Missouri State University
jws086f@smsu.edu

Duane R. Moses
Missouri State University
drm167f@smsu.edu

ABSTRACT

This paper describes the initial efforts to explore the benefits of OO development by describing a framework for categorizing benefits based on phases of system development and by level of analysis. The level of analysis of OO benefits can be the enterprise level, the project level, and the individual developer level. The framework was tested by conducting structured interviews with information systems professionals to assess their perceptions of the reasons for selecting OO technology and OO benefits at the three levels of analysis. The initial results indicate that OO is selected for use for a variety of reasons, but primarily because of perceived productivity benefits. The greatest benefit is thought to be at the enterprise level or project level, with fewer benefits for the individual programmer. Of the benefits often mentioned in the literature, OO being a more natural way of thinking for developers did not appear to be supported.

Keywords

OO development, OO benefits, OO research framework

INTRODUCTION

Object-oriented information system development (OO) is dominating the industry, as noted by the increase of in use of Java and .NET programming platforms for developing new systems. Although OO technology has been available for several decades, it is still not well understood or fully used for business information systems (Cho and Kim, 2002). Reasons for using OO and actual benefits of OO remain controversial (Bryant and Evans, 1994; Hardgrave, 1997). There have been few attempts to systematically hypothesize the benefits of OO development, and empirical measurement of benefits remains problematic.

This paper describes a research effort that develops a preliminary framework for hypothesized benefits, proposing that they potentially occur at three levels – the individual developer, the project team, and the enterprise. Structured interviews with working system development professionals were used to validate the preliminary model. By defining types of benefits and three levels of impact, the study provides the foundation for a research stream that will model and systematically assess OO benefits.

PRIOR RESEARCH

Although OO programming originated in the 1960s, interest in using OO technologies for organizational information systems is fairly recent. Some of the earlier attempts to codify object-oriented systems analysis (OOA) and object-oriented system design (OOD) techniques began in the late 1980s and early 1990s. An early book on OOA proposed object-oriented modeling as a solution to problems ranging from requirements failure, rush to implementation, and faults and inconsistencies in the final system (Shlaer and Mellor, 1988). Edward Yourdon and James Martin, both influential information systems methodologists, published more comprehensive books on OOA and OOD techniques that outlined the key benefits, although no empirical evidence was cited. Coad and Yourdon (1991) discussed three major benefits of object-oriented development: increased productivity, increased quality, and elevated maintainability. Martin (1993) discussed OO as a revolutionary technology with great potential benefits because it provides a more natural way to think about systems and it simplifies complex systems. He went on to list thirty-two specific advantages including reusability, more realistic modeling, faster design, reliability, and easier maintenance.

The most influential OO methodologists since the mid 1990s are Grady Booch, James Rumbaugh, and Ivar Jacobson. Each proposed their own OO development techniques in the early 1990s, and then they combined forces to propose the Unified Modeling Language (UML) and the Unified Process (UP) to attempt to establish *defacto* standard models and techniques for

OO development (Booch, Rumbaugh, Jacobson, 1999; Jacobson, Booch, Rumbaugh, 1999). The assumed benefits of using these models and techniques remained general and anecdotal.

There have not been many academic studies that have addressed advantages and disadvantages of OO development specifically (Briand et al, 1999). Johnson (2000) addressed some controversies surrounding OO development by measuring developer beliefs about the benefits of OO development. Experienced OO developers agreed with the stated advantages and were skeptical about reported disadvantages. Other researchers have focused on specific issues about OO development that are related to benefits. For example, OO is supposed to be a more natural approach, and yet, experienced developers have reported difficulties learning new OO techniques (Sheetz et al., 1997). One study found no advantages in using OO analysis techniques for modeling system requirements compared to using traditional structured techniques (Agarwal et al., 1996), although this study was a test of cognitive fit of the technique to appropriate task. A more recent article attempts to identify the aspects of OO development that require a revolutionary versus evolutionary shift in emphasis to improve the success of OO adoption (Sircar et al., 2001). Again, there is a presumption of benefits to OO development, but the researchers address the reality that moving to OO development remains difficult. A brief review of the literature confirms that the specific benefits of OO development have not been systematically evaluated.

It is clear that the question of the benefits of OO development is a complex construct. For example, the potential benefits differ based on the phase of the project lifecycle being discussed (Bryant and Evans, 1994). There are potential benefits of naturalness and reuse to the analyst and end user when using OO requirements modeling techniques. Research in this context would focus on knowledge representation, requirements elicitation, and analysis patterns, and initial results have been promising (Agarwal, Sinha and Tanniru, 1996; Irwin, 2002). Similarly, there are potential benefits of easier to understand modular architectures and component reuse at the design level (Low, Henderson-Sellers, and Han, 1995; Kim and Stohr, 1998). For the programmer, OO might be difficult to learn initially, but benefits remain in reuse of class definitions and reduced complexity from information hiding (Sheetz et al, 1997; Sanhcez and Choobineh, 1997).

PROPOSED MULTI-LEVEL MODEL

The first task in this study was to develop a framework to more specifically model the aspect of OO benefits being addressed. First, many benefits of OO technology are focused on performance and throughput. These issues are more appropriately addressed by computer science and computer engineering disciplines. Therefore, the first distinction of importance in the framework is to divide benefits of OO technology into OO Performance versus OO Development (see Figure 1). OO Performance addresses issues such as the capability to produce more complex and interrelated systems than possible otherwise and greater efficiency to process larger volumes of data at faster rates. OO Development, on the other hand, addresses the initial development and ongoing support and enhancement of information systems. This research is focused on the benefits of OO Development.

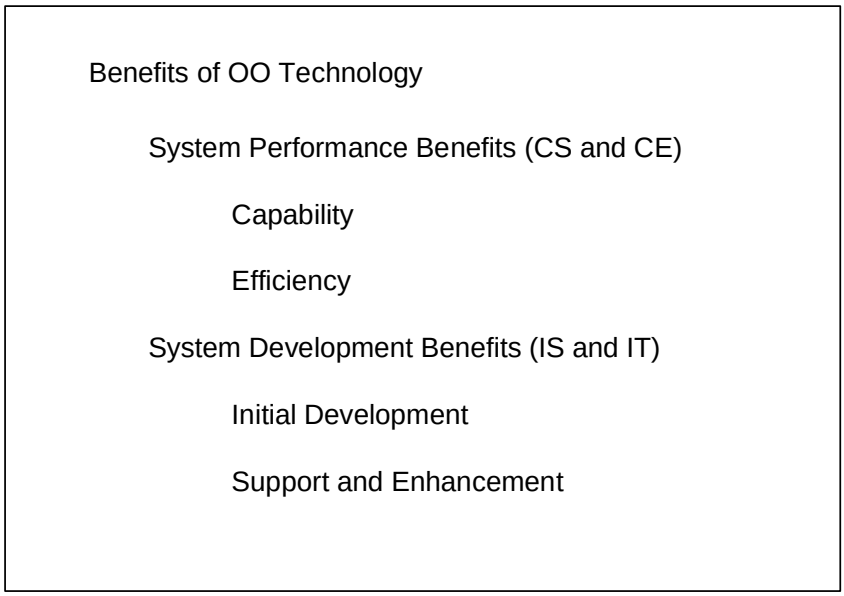

Figure 1. Framework for Benefits of OO at High Level

For OO development, the first distinction of importance is recognizing that benefits accrue at different levels of analysis. For example, some benefits might accrue to the enterprise overall because of OO development. A different set of benefits might accrue to a project team working together on a complex project. Finally, some aspects of OO development might benefit the individual developer working on one part of the project. A second distinction of importance is to recognize that benefits accrue at different phases of the development life cycle, as surveyed in the literature above. Some benefits might apply to defining requirements, while a different set of benefits apply to implementation and testing. Finally, a different set of benefits might accrue for system support and enhancement. Figure 2 shows the framework for describing benefits of OO development at three levels of analysis and with multiple phases of development. Interactions of these dimensions are also implied in the framework, such as enterprise level benefits being potentially different for defining requirements than for implementation and testing.

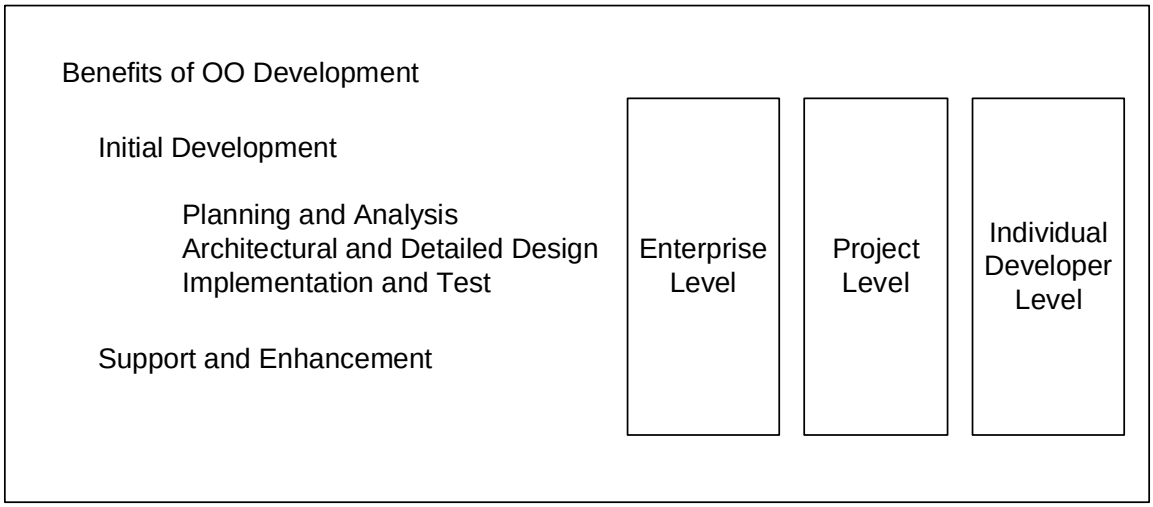

Figure 2. Framework for Benefits of OO Development at Three Levels

RESEARCH DESIGN

This initial study focused on developing a short list of benefits typically mentioned in the literature and using the list to validate the extent to which information systems professionals differentiate OO benefits. The initial focus was on establishing differences across levels of analysis. Differences based on phases of development will be pursued in a later study.

The list of benefits developed for the interviews focused on specific aspects of OO. No comprehensive and exhaustive list of benefits was found that would potentially apply at all three levels of analysis. Therefore, the list was developed by the authors after consulting prior research and current discussion groups on the Internet. Multiple aspects of reuse, naturalness, complexity/modularity, and stability/maintenance were considered. For example, reuse was assessed by asking about error reduction due to reuse and productivity increases due to reuse. Similarly, naturalness was assessed by asking about a natural way of thinking and a more accurate model of the user's world. Each of the benefits could apply at each of the three levels of analysis, although the authors are aware that different meanings and measurement criteria could apply at each level. For example, both naturalness questions could apply to the enterprise level to the extent that requirements can be confirmed in a more natural manner, while naturalness applies to the project team when they communicate and integrate components. Similarly, ease of maintenance allows the enterprise to more rapidly and economically adapt to change, while ease of maintenance for the team or individual developer reduces team and individual effort. The list of potential benefits used in this initial assessment included:

- Reuse reduces errors
- Reuse increases productivity
- More natural way of thinking
- More accurately models users' world

- Easier to maintain
- Encapsulation hides complexity
- More modular - reduces coupling
- Supports component based development
- Design more stable
- Easier to divide work

The framework for benefits of OO development at three levels of analysis was tested by conducting structured interviews with information systems professionals to assess their perceptions of the reasons for selecting OO technology and the benefits of OO technology. Many reasons for use and benefits overlap and are often different views of the same construct. We decided to ask about reasons for use separately from perceived benefits to gain insight into this distinction.

For the most complete picture possible, we decided to ask each interviewee for the three most likely benefits and the three least likely benefits of using OO based on this list. We asked for both most likely and least likely choices for each of the three levels in the framework. Results were captured by the interviewer using a form designed to record the choices. Interviewees were able to read from the lists and point and choose from the form as the interviewer recorded responses. Results of interviews were not expected to result in parametric statistical testing. Rather, the benefits categories and levels of benefits were tested for the development and refinement of an instrument to be used in follow up research.

RESULTS

Twenty-five IT professionals and managers from fifteen moderate to large organizations were interviewed. Fourteen of the fifteen organizations had IT departments staffed with over 30 professionals. These included companies such as Caterpillar, Edward Jones, Cerner, Boeing, FedEx, Sprint, and State Farm. Twenty-three of those interviewed believed there are cost benefits to using OO for system development.

Thirteen of 15 organizations reportedly use OO technology extensively. Of those who do, the most often listed reasons for OO use included the following:

- Increased productivity
- Benefits proven in industry
- Development tools used support OO
- Staff wanting to use OO

The interview sample clearly believed there are productivity benefits to using OO development, and most believe productivity has been proven in industry. It was not clear whether the availability of OO tools and staff interest in OO tools was evidence of proven benefits, however. Those in the two firms that do not use OO mentioned that benefits were ***not*** proven in industry and that there are difficulties integrating with legacy systems as the reasons.

For each of the 25 interviewees, we asked about their specific experience with OO at each of three levels: programmer level, project level, and enterprise level. Five reported no specific OO experience, but results for the other 20 indicated that 2 had experience at all three levels, and 4 had experience at two of three levels. Overall, 11 had experience at the programmer level, 8 at the project level, and 9 at the enterprise level. The programming languages mentioned most often as being used for OO development included 25 mentions of Visual Studio (VB, C#, C++), 18 mentions of Java, and surprisingly 7 mentions of Smalltalk. Because the sample included directors, managers, project leaders, analysts, and application developers, there was a good mix of specific experience.

The main question of interest in this first stage of research is whether respondents can distinguish between benefits at each of three levels of analysis. When ask about where the biggest benefits of OO are achieved, most responses were for the enterprise level (12), next at the project level (8) followed by the programmer level (3). Therefore, the interviewees recognized that benefits accrue to the enterprise and to the project team, but not as clearly to the individual programmer.

Programmer Level Benefits

When interviewees were asked about the three most likely benefits of OO at the individual programmer level, *Reuse increases productivity* received the most responses. At the programmer level, reuse would mean using existing class libraries and previous program code. It could also mean reusing design patterns, reusing user interface forms, and reusing database access classes. The second most responses were for *Reuse reduces errors*, a related benefit that comes from using pre-tested

components. The third most responses were for *Supports component based development*, which also seems to echo the other benefits of reuse. Of the least likely benefits, *More natural way of thinking* received the most responses. It seems clear that OO programming is not considered natural at all by this diverse sample. Interestingly, *Easier to maintain* received mixed results, mentioned many times as likely by some and unlikely by others. Ease of maintenance appears to be controversial at the programmer level. Alternatively, some respondents might have been thinking of the support and enhancement phase as opposed to the implementation and test phase, which would account for the mixed results. Similarly, more accurate user models might apply to planning and analysis more than implementation, accounting for mixed results. Comparison across life cycle phases will be conducted in a follow up study. The frequencies sorted by rank at the programmer level are shown in Table 1. Again, each interviewee was asked to select three most likely benefits and then three least likely benefits separately.

Programmer Level – *Most Likely* Benefits		Programmer level – *Least Likely* Benefits	
Frequency	**Ranked Benefits**	**Frequency**	**Ranked Benefits**
12	Reuse increases productivity	14	More natural way of thinking
11	Reuse reduces errors	10	Encapsulation hides complexity
10	Supports component based development	9	Easier to maintain
9	Easier to maintain	8	More accurately models users' world
5	Encapsulation hides complexity	7	Easier to divide work
4	More accurately models users' world	6	Design more stable
4	Easier to divide work	5	Reuse reduces errors
3	More modular - reduces coupling	2	More modular - reduces coupling
2	Design more stable	2	Supports component based development
1	More natural way of thinking	0	Reuse increases productivity

Table 1: Programmer Level Frequencies

Project Level Benefits

The frequencies sorted by rank at the project level are shown in Table 2. At the project level, *Reuse increases productivity* and *Reuse reduces errors* again appear in the top three most likely benefits. Two other likely benefits, *Easier to divide work* and *Design more stable*, are mentioned often as likely benefits at the project level, but not at the programmer level. This is not surprising as dividing up the work and having a more stable design are important to the project team. This is an example of some bias in the list of benefits, but it is also evidence that the interviewees were attending carefully to the context of the questions. As such, follow research using this approach seems promising. Unlike at the programmer level, *Easier to maintain* is more consistently viewed, mentioned less often as likely and more often as unlikely. Perhaps maintenance was viewed as an individual effort rather than team effort. The most unlikely benefits at the project level were *More natural way of thinking* and *More accurately model users' world*. This indicates that most respondents were probably thinking of the project team as it worked at the implementation and test phase of the project rather than planning or analysis, where naturalness and user models would be more important. Alternatively, respondents might not accept the premise of naturalness given their specific experiences with OO. Similarly, respondents were probably thinking about the project team during initial development rather than during support and enhancement because ease of maintenance was not a clear benefit. It becomes clear that questions about specific benefits should be placed in the context of specific life cycle phases or activities.

Project Level – *Most Likely* Benefits		Project level – *Least Likely* Benefits	
Frequency	**Ranked Benefits**	**Frequency**	**Ranked Benefits**
14	Reuse increases productivity	12	More natural way of thinking
12	Easier to divide work	9	More accurately models users' world
11	Reuse reduces errors	8	Easier to maintain
8	Supports component based development	7	Encapsulation hides complexity
8	Design more stable	6	Reuse reduces errors
6	Easier to maintain	6	Design more stable
5	More modular - reduces coupling	6	Easier to divide work
4	Encapsulation hides complexity	5	More modular - reduces coupling
2	More accurately models users' world	5	Supports component based development
2	More natural way of thinking	19	Reuse increases productivity

Table 2: Project Level Frequencies

Enterprise Level Benefits

At the enterprise level, reuse, ease of maintenance, and support for component based development received the most responses. *More accurately models users' world* is mentioned more often at the enterprise level, perhaps indicating that many respondents were thinking of the final result when considering the enterprise level, such as how well the finished system matched user requirements. *Easier to maintain* is also mentioned more often, indicating that many respondents were thinking about the support and enhancement phases, with *Reuse reduces error* possibly applying to the errors during system operation. On the other hand, *Easier to maintain* and *Reuse reduces errors* are also the least likely benefits mentioned, indicating many respondents were not thinking of on going support and enhancement. Again, it becomes clear that research exploring benefits must place specific questions in the context of a life cycle phase.

Enterprise Level – *Most Likely* Benefits		Enterprise level – *Least Likely* Benefits	
Frequency	**Ranked Benefits**	**Frequency**	**Ranked Benefits**
12	Reuse increases productivity	9	Easier to maintain
9	Reuse reduces errors	8	Reuse reduces errors
8	Easier to maintain	8	More natural way of thinking
8	Supports component based development	8	Easier to divide work
7	More accurately models users' world	7	Encapsulation hides complexity
7	Design more stable	6	More accurately models users' world
5	Encapsulation hides complexity	5	More modular - reduces coupling
5	More modular - reduces coupling	5	Design more stable
5	Easier to divide work	3	Supports component based development
4	More natural way of thinking	1	Reuse increases productivity

Table 3: Enterprise Level Frequencies

CONCLUSION

This paper reports initial research that developed a framework for studying OO benefits and then conducted structured interviews with system professionals to validate the approach taken with the framework. The sample size and question format precluded parametric statistical testing, but a careful analysis of responses showed some trends and confirmed that respondents could differentiate between benefits at the programmer level, project level, and enterprise level. The responses also revealed that questions about specific benefits should be placed in the context of a system development life cycle phase or activity, which will be done in a follow up study. Refinements to the list of benefits used in further research will be guided by this finding. Overall, the framework which focuses on OO development specifically, with three levels of analysis and multiple life cycle phases, appears to be a promising approach to research on the benefits of OO development.

There are several limitations to this study that should be discussed. The sample size is small, although each interviewee was interviewed systematically and given the chance to elaborate and ask questions. The responses are also difficult to separate from the specific context of the respondent, such as organizational role, experience level, type of organization, type of project, and technological environment. The list of benefits and their applicability to all three levels of analysis is also in need of further refinement. Adding specific benefits that can be applied to all three levels plus all life cycle phases will be a challenging and require several iterations. Further work on the specific benefits and on the research framework dimensions, however, will lead to better understanding of this complex subject.

Specific findings from this research indicate that system developers believe that OO development has cost benefits, primarily due to productivity improvements based on reuse. Most assume the benefits have been proven in industry, possibly based on the OO tools available and on developer interest in using OO technology. More studies such those by Kim and Stohr (1998) and Sanhcez and Choobineh (1997) that address reuse are called for. Of the three levels in the framework, the enterprise level followed by the project level are where the OO benefits are thought to accrue. Benefits to the individual programmer are less certain, specifically the claim of naturalness and reduced complexity. More research about learning OO techniques is also called for (Sheetz, et al, 1997). This research and the development of the OO benefits framework should provide insight into areas where empirical research on OO benefits will be the most useful.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers and the mini track chair for their thoughtful and useful comments and suggestions for improving this manuscript and for conducting follow up work.

REFERENCES

1. Agarwal, R., Sinha, A., and Tanniru, M. (1996) Cognitive fit in requirements modeling: A study of objects and process methodologies, *Journal of Management Information Systems*, 13, No. 2, 137-162.
2. Booch, G., Rumbaugh, J. and Jacobson, I. (1999). *The Unified Modeling Language User Guide*, Upper Saddle River, NJ: Addison-Wesley.
3. Briand, L. Arisholm, E. Counsell, S. Houdek, F. and Thevenod-Fosse, P. (1999) Empirical studies of object-oriented artifacts, methods, and processes, *Empirical Software Engineering: International Journal*, 4, No. 4, 387-404.
4. Bryant, T. and Evans, A. (1994). OO oversold: Those objects of obscure desire, *Information and Software Technology*, 36, 1, 35-42.
5. Coad and Yourdon (1991) Coad, P. and Yourdon, E. (1991) *Object-Oriented Design*. Englewood Cliffs, NJ: Yourdon Press/Prentice Hall.
6. Cho, Ihlsoon and Kim, Young-Gul (2002). Critical factors for assimilation of object-oriented programming languages, *Journal of Management Information Systems*, 18, 3 (winter 2002), 125-156.
7. Hardgrave, B. (1997). Adopting object-oriented technology: evolution or revolution? *The Journal of Systems and Software*, 37, 1, 19.
8. Irwin, G. (2002). The role of similarity in the reuse of object-oriented analysis models, *Journal of Management Information Systems*, 19, 2, 219-248.

9. Johnson, R. (2000) The Ups and Downs of Object-Oriented Systems Development, *Communications of the ACM*, 43, No. 10, 69-73.
10. Jacobson, I, Booch, G. and Rumbaugh, J. (1999) *The Unified Software Development Process*, Upper Saddle River, NJ: Addison-Wesley
11. Kim, Y. and Stohr, E. (1998). Software reuse: Survey and research directions, *Journal of Management Information Systems*, 14, 4, 113-149.
12. Low, G., Henderson-Sellers, B and Han, D (1995). Comparison of object-oriented and traditional systems development issues in distributed environment, *Information & Management*, 28, 5, 327-340.
13. Martin (1993) Martin, J. (1993) *Principles of Object-Oriented Analysis and Design*. Englewood Cliffs, NJ: Prentice Hall.
14. Sanhcez, N. and Choobineh, J. (1997). Achieving reuse with OO technology, *Information Systems Management*, 14, 2, 48.
15. Sheetz, S., Irwin, G., Tegarden, D., Nelson, H., and Monarchi, D. (1997) Exploring the difficulties of learning object-oriented techniques, *Journal of Management Information Systems*, 14, No. 2, 103-131.
16. Shlaer, S and Mellor, S. (1988) *Object-Oriented Systems Analysis: Modeling the World in Data*. Englewood Cliffs, NJ: Yourdon Press/Prentice Hall.
17. Sircar, S. Nerur, S. and Mahapatra, R. (2001) Revolution or evolution? A comparison of object-oriented and structured systems development methods, *MIS Quarterly*, 25, No. 4, 457-470.