Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2000 Proceedings

Americas Conference on Information Systems (AMCIS)

2000

Software Reuse in Information Systems Development

Marcus A. Rothenberger
University of Wisconsin - Milwaukee, rothenb@uwm.edu

Uday R. Kulkarni
Arizona State University, uday.kulkarni@asu.edu

SOFTWARE REUSE IN INFORMATION SYSTEMS DEVELOPMENT

Marcus A. Rothenberger, School of Business Administration; University of Wisconsin-Milwaukee, rothenb@uwm.edu
Uday R. Kulkarni, School of Accountancy and Information Management; Arizona State University, uday.kulkarni@asu.edu

Introduction

Software and Management Information Systems application development have become a key area to the performance of most firms. The reuse of previously written code is a way to increase software development productivity as well as the quality of the software (Basili, et al., 1996; Gaffney and Durek, 1989). If previously tested components are reused in a new software project, they are more likely to be error free than new components. This reduces the overall failure rate of the software project. Case studies, such as (Banker and Kauffman, 1991; Poulin, et al., 1993; Apte, et al., 1990; Lim, 1994; Swanson, et al., 1991) were instrumental in obtaining such insights. Today, an increasing number of organizations are adopting the practice of software reuse (Lim, 1994). A common misconception is that object-orientation alone will lead to reuse. While it can help facilitating a reuse approach, research has shown that object technology does not always lead to reuse (Fichman and Kemerer, 1997).

Software reuse requires a substantial up-front investment for the development and maintenance of a software repository with reusable components (Barnes and Bollinger, 1991). A large part of the set-up cost comes from the fact that additional effort is needed to make regular components generic enough for use in future projects (Mili, et al., 1994). In the long-run, this initial investment can be offset by the cost savings through reuse. Using a reusable component in lieu of writing a new component from scratch saves development cost. However, the component has to be located and retrieved from the repository. Often, components cannot be used as is, but also need to be modified to fit the context of the new project. Reuse can only be economically viable, if the savings achieved through reuse will over time offset the start-up cost of implementing the reuse methodology and populating the software repository.

While software reuse is not a new research area to computer science (Krueger, 1992), MIS research has only recently begun to investigate this important aspect of software development. This reflects an increased understanding that too little work has been done on non-technical issues (Zand and Samadzadeh, 1995). Organizational and behavioral aspects, legal constraints, and economic considerations are little explored in the context of software reuse. IS research can also contribute to storage and retrieval problem by developing domain specific solutions. In this overview we summarize the work done in the areas most important to IS research.

Organizational and Behavioral Issues

Research has investigated organizational and behavioral factors that can determine the success of a reuse program. The most frequently addressed issues are summarized below:

Reuse requires a shift in the software development paradigm. This affects the entire IT department or organization and hence cannot be successfully implemented without the full support of high-level management (Frakes and Isoda, 1994; Isoda, 1992; Lee and Litecky, 1997). One specific area where management support is effective is the recognition that developers need to expand on new skills and unlearn old habits. Hence, a successful reuse program requires systematic education for reuse (Frakes and Fox, 1995; Lee and Litecky, 1997).

Reusable software components need to be written in a more generic fashion in order to be suitable for repeated reuse. Hence, it is important for the success of reuse that modules are designed accordingly. (Frakes and Isoda, 1994).

Other issues affecting reuse relate to the commonality of the domain of the software components and to the population of the repository. The more focused the domain is in which reuse is desired, the more is the likelihood of reuse. A focused domain ensures that components are needed across multiple projects (Isoda, 1992; Lee and Litecky, 1997). Furthermore, large-scale reuse is only feasible if the repository holds a sufficient number of components. (Frakes and Isoda, 1994; Lee and Litecky, 1997). This requires a substantial up-front development effort.

An organization wanting to reuse components to which other entities have proprietary rights will face legal constraints. It is important that those are considered, so they don't inhibit reuse (Frakes and Isoda, 1994).

Reuse Measurement

To ensure the financial success of reuse, software developers need to monitor the success of their reuse methodology using metrics (Isoda, 1992). The reuse rate is the most basic reuse measure, however, its assessment is not trivial (Poulin, 1997). Questions as to what counts as reuse and to what extent need to be answered. (Banker, et al., 1994) implemented a set of metrics that assesses the reuse rate in a repository-based CASE environment. (Rothenberger and Hershauer, 1999) presented a reuse rate measure in the context of an enterprise-level data model based reuse environment.

By measuring the performance of projects in a reuse environment the organization sends a message to its employees that reuse is important, and that successful practices will be noticed and rewarded. (Rothenberger and Dooley, 1999) have developed a project performance measure to address this issue.

Another research stream within the reuse measurement area has looked at how to measure the economic aspect of reuse. Economic models help organizations to understand whether investment in reuse pays off. (Gaffney and Durek, 1989) have discussed a return-on-investment model that addresses the issue of how many times a component has to be reused to be paid off. The model by (Barnes and Bollinger, 1991) looks at reuse economics as a cost-benefit tradeoff. (Pfleeger and Bollinger, 1994) introduce a technique where estimates are made by comparing a proposed project with its differences from a baseline project, thus allowing estimation of the impact of reuse over multiple projects.

The reuse measurement research has presented a comprehensive set of metrics that captures the reuse tradeoffs from various aspects. However, the more complex the models are the harder they are to implement in an organizational setting. Many models rely on the availability of parameters that are very difficult to obtain. For example, the cost of developing reusable software relative to that developed without reuse in mind and the additional cost of developing components generic enough for reuse was addressed by (Poulin, 1997). Future research needs to focus on the application of those measures and on developing techniques that allow estimating the hard-to-obtain parameters.

Component Retrieval

Efficient storage and retrieval of components is crucial to the success of software reuse. The complexity of the information associated with software modules makes it difficult to efficiently identify the needed component. (Chen and Lee, 1993) developed a classification scheme for software components similar to cataloging of Integrated Circuits. Efficient retrieval is feasible if the right classification schemes are used. (Latour and Johnson, 1988) took this approach a step further by developing a graphical retrieval system for reusable components.

(Prieto-Díaz, 1991) present software classification principles that employ a classification method similar to that used in library science. (Rothenberger and Hershauer, 1999) have described a reuse environment where the reusable components are mapped to the business process and data model based on the areas of interaction. This example demonstrates that component retrieval can work efficiently within a highly specialized domain.

The approaches discussed in the literature only represent partial solutions to the retrieval problem. In most cases where reuse is common, developer's experience with and knowledge about the repository is crucial.

Conclusion and Future Directions

Organizational factors such as management support, importance of reuse-related skills training explain only part of the variation in software reuse rates across organizations. Future studies may need to look into project-level issues such as precedence and sequencing, urgency of delivery, etc. Clients who outsource information systems development also play a substantial role in determining the extent of reuse in their projects.

In the measurement arena, we may mention that present economic models do not consider the benefits of reuse that go beyond monetary savings. Reuse can create a competitive advantage through higher quality software and an improved ability of the organization to respond to change requests. Quantitative models cannot entirely capture the cost-benefit tradeoff of software reuse. Capturing such data is crucial to management of reuse.

The ideal retrieval method would neither require any prior knowledge of the repository nor any informal communication among developers to find the components needed. The approaches discussed in the literature take steps towards such an ideal retrieval method, but still rely on informal ways of finding the right component. Future research must strive towards offering complete solutions to the storage and retrieval of components.

Many organizations have invested in Enterprise Resource (ERP) systems for standardizing their traditional transaction processing needs. ERP systems exhibit a high degree of reuse capability for such systems. In spite of these shifts, customized development of specialized components will constitute the bulk of the systems development efforts in the near and mid-term future. With the skyrocketing development costs of information systems and their critical nature in today's businesses, the reuse issues highlighted here will continue to be of importance to both the research and practitioner community.

References

Apte, U., Sankar, C.S., Thakur, M. and Turner, J.E. "Reusability-Based Strategy for Development of Information Systems: Implementation Experience of a Bank," *MIS Quarterly* (14:4), 1990, pp. 420-433.

Banker, R.D. and Kauffman, R.J. "Reuse and Productivity in Integrated Computer-Aided Software Engineering: An Empirical Study," *MIS Quarterly* (15:3), 1991, pp. 374-401.

Banker, R.D., Kauffman, R.J. and Zweig, D. "Automating Output Size and Reuse Metrics in a Repository-Based Computer Aided Software Engineering (CASE) Environment," *IEEE Transactions on Software Engineering* (20:3), 1994, pp. 169-187.

Barnes, B.H. and Bollinger, T.B. "Making Reuse Cost-Effective," *IEEE Software* (8:1), 1991, pp. 13-24.

Basili, V.R., Briand, L.C. and Melo, W.L. "How Reuse Influences Productivity in Object-Oriented Systems," *Communications of the ACM* (39:10), 1996, pp. 104-116.

Chen, D.-J. and Lee, P.J. "On the Study of Software Reuse Using Reusable C++ Components," *Journal of Systems Software* (20), 1993, pp. 19-36.

Fichman, R. and Kemerer, C. "Object Technology and Reuse: Lessons from Early Adopers," *IEEE Computer* (30), October 1997, pp. 47-59.

Frakes, W.B. and Fox, C.J. "Sixteen Questions About Software Reuse," *Communications of the ACM* (38:6), 1995, pp. 75-91.

Frakes, W.B. and Isoda, S. "Success Factors of Systematic Reuse," *IEEE Software* (11), September 1994, pp. 15-19.

Gaffney, J.E. and Durek, T.A. "Software reuse - key to enhanced productivity: some quantitative models," *Information and Software Technology* (31:5), 1989, pp. 258-267.

Isoda, S. "Experience Report of Software Reuse Projects: Its Structure, Activities, and Statistical Results," *Proceedings of the Proceedings of the 14th International Conference on Software Engineering*, Melbourne, Australia, 1992, pp. 320-326.

Krueger, C.W. "Software Reuse," *ACM Computing Surveys* (24:2), 1992, pp. 131-183.

Latour, L. and Johnson, E. "Seer: A Graphical Retireval System for Reusable Ada Software Modules," *Proceedings of the Third International Conference on Ada Applications and Environment*, Piscataway, NJ, 1988, pp. 105-113.

Lee, N.-Y. and Litecky, C.R. "An Empirical Study of Software Reuse with Special Attention to Ada," *IEEE Transaction on Software Engineering* (23:9), 1997, pp. 537-549.

Lim, W.C. "Effects of Reuse on Quality, Productivity, and Economics," *IEEE Software* (11:5), 1994, pp. 23-30.

Mili, H., Witt, J., Radai, R., Wang, W., Strickland, K., Boldyreff, C., Heger, J., Scherr, W., Olsen, L. and Elzer, P. "Practitioner and SoftClass: A Comparative Study of Two Software Reuse Research Projects," *Journal of Systems and Software* (25), 1994, pp. 147-170.

Pfleeger, S.L. and Bollinger, T.B. "The economics of reuse: new approaches to modeling and assessing cost," *Information and Software Technology* (36:8), 1994, pp. 475-484.

Poulin, J.S. *Measuring Software Reuse: principles, practices, and economic models*, Addison-Wesley, Reading, MA, 1997.

Poulin, J.S., Caruso, J.M. and Hancock, D.R. "The business case for software reuse," *IBM Systems Journal* (32:4), 1993, pp. 567-594.

Prieto-Díaz, R. "Implementing Faceted Classification for Software Reuse," *Communications of the ACM* (34:5), 1991, pp. 88-97.

Rothenberger, M.A. and Dooley, K.J. "A Performance Measure for Software Reuse Projects," *Decision Sciences* (30:4), 1999,

Rothenberger, M.A. and Hershauer, J.C. "A Software Reuse Measure: Monitoring an Enterprise-Level Model Driven Development Process," *Information & Management* (35:5), 1999, pp. 283-293.

Swanson, K., McComb, D., Smith, J. and McCubbrey, D. "The Application Software Factory: Applying Total Quality Techniques to Systems Development," *MIS Quarterly* (15:4), 1991, pp. 566-579.

Zand, M. and Samadzadeh, M. "Guest Editors' Corner: Software Reuse: Current Status and Trends," *Journal of Systems and Software* (30), 1995, pp. 167-170.