Systems Development with Systematic Software Reuse: an Empirical Analysis of Project Success Factors

Marcus A. Rothenberger
School of Accountancy and Information Management, Arizona State University
(marcus.rothenberger@asu.edu)
Betreuer: J. C. Hershauer, Arizona State University

1 Research Motivation

Systematic reuse of previously written code is a way to increase software development productivity as well as the quality of the software (Gaffney/Durek 1989; Banker/Kauffman 1991; Basili et al. 1996). 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. Hence, high quality software can be built by incorporating reused components into the software project. Companies employing a reuse methodology have to engage in three major activities: the reuse, the retrieval, and the production of reusable software components. While there has been some examination of the organizational factors that make companies succeed or fail in their reuse attempts (i.e., Frakes/Fox 1995; Frakes/Isoda 1994), there is a need for a more rigorous, empirically-based study of these issues at the level of the individual project.

The purpose of this dissertation is to explore what factors can lead to success or failure of employing reuse in projects of software firms that have an overall successful reuse methodology in place. First, a software firm that employs systematic reuse using an enterprise-level model based approach has been selected. To assess project reuse success, a metric has been developed to calculate the reuse rate in such a development environment. One of the software firm's clients has been selected to gain in-depth knowledge as to what factors lead to project reuse success; thus, the unit of analysis is the project, controlling for firm and client differences. Potential factors affecting reuse will be collected as perceptual data from the personnel involved in the projects. The study uses field data and qualitative research methods to explore to what extent the identified factors influence project reuse success. The research concludes by developing a taxonomy of reuse methodologies employed in the industry. The classification is needed to explore the applicability of the propositions developed from the field re-

search to methodologies different from the one observed in the case study. The taxonomy will be established using a survey of companies that employ systematic reuse for their software development. A follow-up study then will test the applicability of the project success factors to each reuse methodology identified in the taxonomy.

2 Systematic Software Reuse

The productivity gain from reuse comes from integrating previously written components into new software projects. It is assumed that additional development effort is required to make a software component generic for reuse. The development of components suitable for reuse normally takes more effort and resources than the development of components customized for one project. The effort spent on retrieving components and the additional effort spent on writing reusable components compared to writing non-reusable components can offset the gains from reuse. Consequently a firm investing in a reuse methodology for its software development needs to monitor the benefit obtained from reuse. Only if the benefit from reusing components is higher than the cost of implementing and maintaining the reuse driven development methodology can the approach be considered successful. However, the benefit can only partially be measured in terms of cost savings. Benefits like quality improvement and quick response time are qualitative and difficult to precisely determine. We can use the reuse percentage as a measure of reuse success on the project level.

The traditional method of component reuse is the use of a software repository-based environment. Components are retrieved using tools such as browsers, which match specifications and documentation of the components against the requirements of the project. Another approach for dealing with the component retrieval problem is the use of an enterprise-level model to facilitate software reuse in custom business software development (Rothenberger/Hershauer, 1998). The retrieval of components in such an environment is easier than in a component library environment, since most components are part of the data/process model which becomes an integrated part of the software project. The decision about what parts of the data model to use for a project ultimately leads to the decision about what code components to reuse. Choosing a company for this research that employs an enterprise-level model based methodology provides an environment for potentially successful reuse.

3 Research Method

3.1 The Research Questions

1. *How can an established reuse rate measure be applied to the enterprise-level software development context to assess the percentage of development effort reused?*

2. *What are the factors that affect the reuse success of software projects developed with systematic reuse in an enterprise-level model context and how do they affect success?*

3. What is the comprehensive set of reuse methodologies used for software development by industry and what features are differentiating those methodologies?

3.2 First Research Question

3.2.1 Purpose

The literature has not established how to calculate the reuse rate in an enterprise-level model driven software development environment. The purpose of answering this research question is to ascertain that an established reuse measure can be applied to such an environment and to present an approach for developing the measure. The reuse rate is a necessary factor for the assessment of the value of the reuse method to a software development firm. It is essential to assess the reuse rate to monitor the success of the software reuse methodology. According to Poulin (1997), the de-facto standard for measuring the reuse level is:

$$\frac{\mathrm{Re}\,used\_Software}{Total\_Software} \times 100\% .$$

More detailed methods that have been published are all based on this general notion of software reuse (i.e., Basili et al. 1996; Frakes/Terry 1996). The methods differ in what they consider reused software and how they measure the "amount" of software. They extend the general notion of software reuse to a defined approach which allows data collection in actual software projects. However, the established reuse metrics can only be applied to an environment where the reused components are easily identifiable, like in a software repository-based development environment. Other reuse paradigms, like object-orientation or enterprise-level models, are frequently employed to facilitate reuse. While Basili et al. (1996) have applied reuse metrics to an object-oriented context, nothing has been presented for calculating a reuse percentage in an enterprise-level software reuse context.

3.2.2 Method

We answer this question by establishing an approach to a metric for the measurement of reuse in a generic enterprise-level model context and by using this approach to create a specific metric for a company. Traditionally, the reuse rate is defined as the percentage of the development effort retrieved as code segments from a software repository and employed as components for a program, with the software repository containing previously written code. The metric proposed extends this definition to also include development effort reused through employment of a generic enterprise-level model (Rothenberger/Hershauer 1999). An example is given for the successful assessment of a reuse percentage for a software developer's actual project. The approach to the metric is general and can be used to establish a concrete metric for other firms' enterprise-level model-based development environments.

3.2.3 Contribution

The main contribution of answering this research question is the establishment of an approach to measure software reuse in an enterprise-level development context. This is achieved by defining what is counted as reuse and how to count it: A code measure is established that is a good representation of development effort. Pragmatic considerations must be made as to how difficult the measure is to obtain. For the present research, lines of code (LOC) is a good surrogate measure for development effort. As discussed earlier, in an enterprise-level model context software is developed in multiple layers. The reuse rate is calculated separately for each layer. Reusable components that have been modified less than a certain threshold are counted as reuse when integrated into the project. Modules that have been modified more than the threshold are counted as developed from scratch. Basili et al. (1996) use 25 percent of development effort as a reasonable threshold. The second (or higher) reuse of a component is not a benefit of software reuse, since one component can be used several times in a project, even in a non-reuse context. Hence, we count multiple reuse only once towards the measure. Finally, the reused code measure and the total code measure are collected for each of the structures for hand-coded and generated code. The ratio of the two totals is the reuse rate.

We illustrate the implementation of the metric in the context of a specific software development company. The reuse rate is calculated for a particular project of that company to illustrate the feasibility of the method presented. The reuse rate measure developed is used to assess project reuse success in the next part of the dissertation. The reuse rate is generally used to monitor the success of a reuse method and is needed to calculate cost savings achieved through reuse using economic models. The approach for the calculation of the reuse rate in an enterprise-level model driven software reuse context developed in this part of the research enables us to make those assessments for this type of systematic reuse.

3.3 Second Research Question

3.3.1 Purpose

Previous research has explored what makes reuse methods succeed or fail (i.e., Frakes/Fox 1995; Frakes/Isoda 1994). Findings considered companies and methodologies as a whole. The success factors previously identified include organizational reuse experience, tool support, level of management support, level of metrics used, incentive systems, and development standards. Although those organizational factors are constant within one development setting, reuse success across projects varies. This suggests that there is an additional set of factors at the project level as opposed to the organizational level, that affect reuse. To identify the project-level factors, we conduct a field study analyzing quantitative and qualitative data from five projects within a single firm that employs an enterprise-level model for development with reuse. The company specializes in the development of business process and accounting systems.

3.3.2 Method

We are conducting an in-depth case study on a single client's projects of the subject company to identify the factors that influence project reuse success. Project success is assessed for five projects by calculating the reuse rates according to the approach developed in phase one. There is a difference in success across the client's projects in terms of reuse percentage achieved. To explore what factors cause the difference in reuse success across the projects, we identify at least two people for each project that can make valid assessments as to what factors could have contributed to the project's success. Several participants have worked on the same project. We have conducted interviews with the head of software development to identify the respective project managers or head designers. The collection of the perceptual data is conducted in two steps:

1. The developers identify candidate factors for project reuse success.
2. The developers make an assessment of the value for each of the candidate factors in the context of their respective project(s).

To obtain the data for step 1, we use a Nominal Group Technique (NGT) session to facilitate the collection of relevant factors among the project managers and head analysts. We are dealing with a group of eight people. In the idea-generation stage of the meeting, we collect the candidate factors for project reuse success as proposed by the participants. Multiple factors named by different participant may refer to the same construct. The participants will then eliminate the duplicate phrases that are naming the same construct during the NGT session. The developers identified 11 factors that can be grouped into four areas: client concerns, company culture, developer experience, and project attributes.

In step 2, the assessments of the values for the factors will be made through structured interviews with a limited response scale (five point scale) by the developers. A limited scale enables us to recognize patterns when analyzing the data. The participants will be asked to assess the values for the candidate factors with respect to their project(s). To eliminate project performance differences because of the nature of the client, we chose a series of projects conducted with a single client. Hence, we need to anchor the terms in the questions relative to the other projects for this client.

For the final analysis, results will be grouped by the total reuse rate, as well as the reuse rates of each of the three structural levels. The relationship between the factors and the reuse success of each project will be explored. We are looking for within-group similarities coupled with intergroup differences. This search for cross-case patterns allows us to avoid premature or false conclusions as a result of information-processing biases (Eisenhardt 1989). Examining the success for each structural level separately will allow us to conclude where to direct additional effort and investments for improvements. Not every factor has an equal effect on the reuse percentage on all structural levels. Preliminary analysis indicates that client concerns, company culture, and project attributes show patterns. Hence, we conclude that client support of reuse, the internal support of reuse, and the project sequence and therefore the fit of the repository are important for reuse success. The factors dealing with developer experience do not show any patterns. This is consistent with the philosophy of systematic

software reuse, saying that a structured methodology for reuse does not require extensive knowledge of the repository from the developers. While the NGT session suggested that developer experience matters, the patterns show that it does not matter for reuse success.

Follow-up interviews will be conducted to fill in the details to obtain a complete picture of the cases. The result is a model that suggests the factors leading to variations in project reuse success within one development environment. In addition to the perceptual data, we will conduct additional on-site visits to collect observational data that will consist of archival documents; such triangulation facilitates reliability and generalizability. Write-ups will be developed from the observational data identifying the factors affecting project reuse success. They will be shared with the participants and corrections will be made.

3.3.3 Contribution

Identifying the factors that influence project success in a systematic reuse development environment will allow software development firms to better target reuse investments. Previous research has focused on the critical success factors of reuse methodologies on an organizational level. However, sound methodologies may not be able to achieve their full potential because of implementation factors at the project level. Those factors have widely been ignored so far. The outcome of this stage will be a set of propositions forming a theoretical model that attempts to explain why project reuse success varies in an overall successful systematic software reuse setting. Information rich case studies in a one company setting are conducted to develop the propositions. At this stage of the research, the theory applies only for similar development settings – mainly the enterprise-level model-based reuse approach.

3.4 Third Research Question

3.4.1 Purpose

We have identified a set of factors that affect project reuse success in an enterprise-level model reuse setting. A taxonomy of reuse methodologies is needed to explore to what extent findings are generalizable to other systematic reuse-driven software development environments. The taxonomy will be established by analyzing what differentiates systematic reuse methodologies and what those methodologies are. This part of the dissertation research consists of surveying organizations that employ systematic reuse for their software development to empirically explore how companies are facilitating reuse. From the survey results we will develop the taxonomy of reuse methodologies. The applicability of the project reuse success factors to each of the taxonomy's methodologies will be explored in a follow-up study.

3.4.2 Method

Reuse approaches can vary along various dimensions. The literature has identified features of reuse approaches that vary across company settings. Such features include but are not limited to the type of repository employed, the type of domain, the focus on a particular stage of the software life cycle, and the type of tool support. The objective of this part of the research is to empirically survey the features and to group the results into clusters that form distinct systematic reuse methodologies. Among those methodologies we expect the enterprise-level model based approach to be one. The survey questions ask for an assessment of the value of each feature as identified by the literature. The participants are the employees in charge of software development in companies that are developing software with systematic reuse. Factor analysis will be performed to summarize the information obtained in the survey into a limited number of factors. The results are the underlying constructs of the different methodologies. The constructs identified will allow us to identify a taxonomy of methodologies for systematic reuse that describes the set of approaches in use.

References

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

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

Eisenhardt, K. M. (1989): Building Theories from Case Study Research. Academy of Management Review 14(4): 532-550.

Frakes, W./Terry, C. (1996): Software Reuse: Metrics and Models. ACM Computing Surveys 28(2).

Frakes, W. B./Fox, C. J. (1995): Sixteen Questions About Software Reuse. Communications of the ACM 38(6).

Frakes, W. B./Isoda, S. (1994): Success Factors of Systematic Reuse. IEEE Software: 15-19.

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

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

Rothenberger, M. A./Hershauer, J. C. (1998): Facilitating the Retrieval of Reusable Software Components using an Enterprise-Level Model. Proceedings of the Association for Information Systems Americas Conference, August 1998.

Rothenberger, M. A./Hershauer, J. C. (1999): A Software Reuse Measure: Monitoring an Enterprise-Level Model Driven Development Process. Information & Management (forthcoming).