Project-Level Reuse Factors: Drivers for Variation within Software Development Environments

Marcus A. Rothenberger
School of Business Administration, University of Wisconsin–Milwaukee, PO Box 742, Milwaukee, WI 53201, e-mail: rothenb@uwm.edu

ABSTRACT

Systematic reuse can dramatically improve software development productivity and quality even though a software reuse methodology may require substantial investments. Some projects may fail to achieve the targeted amounts of reuse within organizations that are overall successful in employing reuse. To explain such variation, this research explores the effects of project-level factors in the success of software reuse. A model that relates project factors to project reuse success is developed using an information-rich case study approach. The results are based on the insights obtained in a nominal group technique session, triangulated with structured interviews and comparative case studies. Success factors identified by the study relate to client influence, project culture, project attributes, and developer reuse experience. An organization that can successfully identify the factors affecting potential software reuse will be able to better target investments for the improvement of its reuse methodology and thus positively affect its software development productivity and quality.

Subject Areas: Case Study, Components, Computer Science, Management Information Systems, Nominal Group Technique, Productivity, Quality, Software Reuse, and Systems Development Methodologies.

INTRODUCTION

Efficient development of software mandates a reduction of development time while maintaining high quality levels. One way in which the cycle time of a product can be reduced is to minimize the amount of “newness” contained in it (Griffin, 1997). Innovation newness is reduced by having a design that incorporates already existing system components. In the context of software development, this practice is called software reuse. The literature has established that the reuse of software artifacts is an effective way to increase software development productivity (Banker &

*I would like to thank MBA Technologies for providing the case setting for this research and the numerous employees who participated in the study. In particular, I would like to thank Ivonne Cox, Vijay D’Souza, Denise Gaffney, Gary Hall, Brad Kerr, Dave Monfore, Dan O’Neill, Jim Pennington, Dave Schweizer, Michael Tress, and Brandon Ulrich for their time and their input to this study. Further, I would like to thank the vice president for research and the Graduate College at Arizona State University for financial support, as well as James Hershauer, Uday Kulkarni, and Kevin Dooley for their suggestions and comments during the conduct of this research.
Project-Level Reuse Factors

Kauffman, 1991; Chen & Lee, 1993; Gaffney & Durek, 1989; Jeffrey & Lawrence, 1985). Reuse is also improving software quality, because software incorporates previously tested components that are less error-prone than newly developed components (Basili, Briand, & Melo, 1996). Component-based software can be easily modified if requirements change; thus reuse also improves maintainability of the final product (Rombach, 1991). Likewise, if reuse significantly reduces development time, benefits from being quick-to-market can be obtained (Lim, 1994).

Productivity gain from reuse can be achieved by integrating previously written components into new software projects. The development of components suitable for reuse takes more effort and resources than the development of components customized for one project. Reuse also requires an extensive up-front planning effort to match requirements to available components. Thus, time spent on planning for reuse and retrieving components, and additional effort spent on writing reusable components compared to writing nonreusable components, reduces the benefit. Hence, a firm investing in a reuse methodology for its software development needs to monitor the benefit obtained from reuse (Rothenberger & Dooley, 1999). The reuse approach can only be considered financially viable if the benefit from reusing components is higher than the cost of implementing and maintaining the reuse-driven development methodology. Although the benefits of systematic software reuse can be high, most organizations lack success in implementing systematic reuse to its full potential. Most inhibitors are of a managerial rather than a technical nature. While the past literature has focused mainly on technical problems (Zand & Samadzadeh, 1995), the recent research is also addressing the organizational issues. An important research stream in the managerial reuse literature is the investigation of reuse success factors.

Prior research has investigated the role of reuse success factors. The findings of the eight publications that directly address this issue were synthesized and a comprehensive set of reuse success factors was developed (Table 1). These publications pursued various methods to identify the reuse drivers. Early reuse success factor research (Apte, Sankar, Thakur, & Turner, 1990; Incorvaia & Davis, 1990) focused on management support and the integration of reuse in the development process. Incorvaia and Davis (1990) have also stressed the need for intercomponent standards, which later was confirmed by other publications (Biggerstaff, 1992; Davis, 1994; Sonnemann, 1996). Biggerstaff (1992) introduced several new success factors, including domain issues and infrastructure considerations. More recent research expanded the set of success factors even further: Frakes and Fox (1995), who have surveyed software engineers, managers, and educators in 29 organizations, have introduced the notion that different industries have different reuse capabilities. They have also discussed the need for developer skills: the study identified training and education for reuse as a success factor. Other studies have found developers’ reuse experience to be important (Lee & Litecky, 1997; Davis, 1994). Frakes and Isoda (1994) stressed that reuse of software artifacts raises certain legal issues that may inhibit reuse. They also made the case for reuse measurement, which allows an organization to monitor the success of its reuse approach. Most recent reuse success research listed factors that relate to asset availability or asset commonalities (Davis, 1994; Frakes & Fox, 1995; Frakes & Isoda, 1994; Lee & Litecky, 1997; Sonnemann, 1996). Ravichandran (1999) developed and
Table 1: Literature-identified reuse success factors.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Support</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Recognition Awards, Incentives</td>
<td></td>
<td>++</td>
<td></td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Asset Commonalities across Products</td>
<td></td>
<td>++</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Domain Focus, Common Architecture across Product Line</td>
<td></td>
<td>++</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Reuse Experience</td>
<td></td>
<td>++</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Domain Knowledge</td>
<td></td>
<td>++</td>
<td></td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Reuse Education/Training</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Type of Industry</td>
<td></td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Domain Stability</td>
<td></td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Economies of Scale in Market</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Availability and Quality of Assets</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Reuse Measurement</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Integration of Reuse in the Development Process/Reuse Policy</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Intercomponent Standards</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Domain Eng./Design for Reuse</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Product-Line Approach</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Infrastructure Support</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Reuse Implementation</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Technologies</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Economies of Scale in Technologies</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Legal Problems</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Perceived Economic Feasibility of Reuse in the Organization</td>
<td>√</td>
<td>+</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

Empirically evaluated four theoretical models on the interrelationship between reuse dimensions and systems delivery performance. The research found that a reuse process must include incentives and recognition awards for successful reuse, which is consistent with earlier findings by Apte et al. (1990). It also confirmed earlier notions of process integration of reuse.

While the factors identified by these studies vary, the research design of the studies limits the findings to those reuse success factors that affect the reuse capability of the development environments as a whole. However, in a given development
environment, projects can experience different levels of reuse success. Because of this consequence of the design of prior research, such variation in project success cannot be explained by the literature-identified factors. The current research proposes that only some of the established factors affect reuse success equally across projects in one development environment. In Table 1 those factors are marked as “Constant across Projects.” The other factors may be part of a new category of factors that explain why projects in otherwise capable reuse environments may fail to achieve the targeted amount of reuse. As past research has not explored project-level reuse factors, we do not expect that the existing list of nonconstant factors is comprehensive; thus there is a need to investigate the set of factors that affects reuse success on the project level. A case study is conducted that captures the knowledge of practitioners and compares it to project data. This research contributes to the existing body of knowledge by explaining project-level reuse variation with an extended set of success factors that are investigated on the project level.

The paper is structured as follows. First, the stage for the research is set and the research model is explained. Then the field setting is described. The research methods are explained and the results of each step of the analysis are presented. Finally, the limitations of the study and a summary of the findings are discussed.

THE RESEARCH MODEL

This research aims to identify project-level reuse success factors by conducting comparative case studies analyzing multiple reuse projects. The case study method allows obtaining a good understanding of the nature of project-level reuse factors. It is an appropriate way to study this new aspect of reuse success factors with the objective to extend existing theories (Benbasat, Goldstein, & Mead, 1987). The set of reuse success factors identified in prior research (Table 1) serves as a foundation for the analysis. Most of the literature-identified factors do not vary across the software projects of the case setting; hence they cannot explain why in a given organizational environment, certain projects achieve their reuse objectives better than other projects do. On the other hand, the values of some of the success factors can change across projects. Although prior research has only presented them in the context of an organizational reuse capability assessment, they must be considered as candidates for the set of project-level factors that is to be developed in this study. This process ensures that no previously identified success factor will be accidentally omitted from the findings. However, the research does not stop at reevaluating existing success factors, but also aims to identify project-level factors not considered previously.

Figure 1 presents a research model that is based on the literature findings about reuse success. Prior research has established that organizational factors, together with the reuse methodology employed and the size of the software component repository, determine the organization’s reuse capability (Frakes & Fox, 1995; Frakes & Isoda, 1994; Isoda, 1992; Lee & Litecky, 1997; Ravichandran, 1999; Rine & Sonnemann, 1998). Assuming that the cost for the reuse methodology is held constant, an increased reuse level in a project will result in a reduction of development cost, an increase of software quality, and an improvement in maintainability (Basili et al., 1996; Gaffney & Durek, 1989). This research proposes that not only
Figure 1: The research model.

The organizational reuse capability, but also the set of project-level factors, determine reuse success for each project. The dotted frame on the diagram depicts this extension of the established theories with the notion of project-level reuse success factors. This study will identify these project-level factors and explore their effect on reuse.

THE FIELD SETTING

Selecting projects that have been carried out by one software development company for one client organization eliminates the confounding effects of uncontrollable variables. Factors such as differences in development methodology and differences in the nature of the customer are held constant. Thus, a one-company case study ensures that the variations in reuse success observed are attributed to factors other than organizational reuse success factors, while also allowing meaningful and generalizable results (Apte et al., 1990; Banker & Kauffman, 1991). The
company must have a high reuse capability to ensure that at least some successful reuse projects are included in the study. Hence the selection criterion is that the organization scores high on the organizational reuse success factors.

A medium-size software development and consulting firm headquartered in Phoenix, Arizona, MBA Technologies, was selected for the study. Management has agreed to have the organization’s identity disclosed in the study. The company specializes in the development of business process and supply chain systems. At the time of the study, thirty software engineers were working on custom software development. The organization was founded on, among other principles, the idea of software reuse, and therefore never faced the problem of introducing reuse to an established development process. Thus, the company has successfully employed software reuse in most of their projects for a period of more than ten years.

Prior research has identified factors that determine an organization’s reuse potential (marked in Table 1 as “Constant across Projects”). Thus, the company’s reuse capability was assessed by obtaining values for these factors in a series of interviews with employees and management. This evaluation revealed that MBA Technologies has a high potential for success in reuse according to most values of the organizational reuse success factors relating to the development process, company characteristics, and the internal support of reuse by management and employees. The organization scores high on all factors with the exception of

- industry type (the software industry has only medium success in reuse)
- measurement of reuse (there is no measurement program in place).

**Reusable Components**

Efficient retrieval is an important issue for the success of reuse (Krueger, 1992). MBA Technologies has recognized this and uses a process in which code is retrieved through a naming convention that maps components to a generic business process and data model. The organization has classified their code into three levels of abstraction: low, medium, and high. Low-level components are modules that directly access the information associated with the data model (e.g., modules that update fields, execute processes, and so on). Medium-level components access the data model by calling the low-level components (e.g., calculation routines and more complex algorithms). High-level components are mostly custom parts that are entirely written from scratch (e.g., modules specific to the organizational process). The high-level components call the medium- as well as the low-level components during their execution. Both medium- and high-level components do not directly access the database. They only interact with the data model through the call of low-level components. The organization focuses on the reuse of low-level code, as the reuse methodology employed best supports the retrieval of low-level components. Less reuse is obtained from medium- and high-level code, since the organization relies on less-effective retrieval methods (i.e., keyword and manual searches) for those types of components. The emphasis on low-level code reuse is consistent with Apte et al. (1990), who found that reuse of lower-level program fragments is much easier to facilitate and tends to be more widely used by programmers.
The Projects
Five projects released during a period of 18 months were investigated. They represent the five most recent projects conducted for a major steel producer with sizes varying from 57 KLOC (thousand lines of code) to 143 KLOC. The projects are order capture, trials, demand planning, invoicing, and incentives. The teams working on these projects were largely composed of different individuals. There were four different project leaders on the five projects. Some team members were working on multiple projects.

The scope of the study is reuse in the context of custom commercial software projects, so the projects had to be nontrivial; in the interviews it was determined that three or more analysts were involved in each of the selected projects over a period of several months. It was confirmed that the projects address different stages of the supply chain cycle from order capture to invoicing, representing the variety of projects typical for the software development company. Hence, internal validity can be warranted at this stage. The customer relationship is of a cooperative long-term nature and typical for the industry. Such a commitment is beneficial for achieving high reuse over a series of projects. Since this is a similar situation to what software development groups in large organizations face with their in-house clients, the findings of this study can apply to such an environment as well as to software development companies with external clients. Table 2 provides additional information about the projects.

RESEARCH METHODS
The study triangulates the results of three steps: a review of the literature in combination with a nominal group technique (NGT) session, a comparative case study, and follow-up interviews.

The first step of the analysis identifies an initial set of project-level reuse success factors employing a review of the literature and results of the NGT session conducted with eight software designers and developers of whom most were also involved in the projects. In the second step, evidence obtained from the analysis of five projects using quantitative and qualitative data produced a list of factors that affected reuse success. In the third step, structured interviews with the participants in the NGT session and one management representative helped explain how and why these factors were associated with reuse. The initial set of factors was compared to the findings of the project analyses and the results of the structured interviews (Figure 2). The observations were explained with the knowledge acquired during these interviews.

Identification of Candidate Factors
The research began by identifying a set of candidate project-level factors from the reuse success factors that were presented in the literature in an organizational context (Table 1). Factors that by their nature do not change values across the projects of the field organization were eliminated from further consideration. Such constant factors include type of industry, domain stability, reuse education/training, and so on (the complete set of excluded factors is marked in Table 1). The remaining
Table 2: Projects.

<table>
<thead>
<tr>
<th>Order Capture</th>
<th>This was the first project of the series. At the time of the development there was only a legacy system for building orders that had been developed by a different software development company. Reuse has been obtained from an order capture system that previously had been built for another customer. The reuse rate (percentage of development effort reuse from prior projects) of 50.5 percent is the second lowest among the five projects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trials</td>
<td>Trials enables the organization to carefully track an order that goes through the manufacturing process. Once the trial is successful, it will be turned into a real order. With 76.0 percent reuse, this project achieved the highest reuse level of the five.</td>
</tr>
<tr>
<td>Demand Planning</td>
<td>Demand planning provides the tool set to manage order promising. When a customer of the company that is using the software places an order, the company can predict the earliest day at which they can commit to produce the product in the quantity required. The reuse rate achieved in this project is 67.6 percent.</td>
</tr>
<tr>
<td>Invoicing</td>
<td>Invoicing takes data from a shipping system, looks at what has been produced, determines the list of materials and generates the invoice and the invoice detail. It also prints the invoices and prepares them for mailing. The reuse rate of this project is 67.4 percent and lies, together with Demand planning, in the middle among the five projects.</td>
</tr>
<tr>
<td>Incentives</td>
<td>This project calculates a pay incentive for groups of employees based on business measurements such as productivity and consumption of materials. This project is in the domain of decision support and performance measurement and is therefore different from the previously developed operation- and transaction-driven systems. The interaction with the other systems is limited to read-only access to common operational data. The first release included three incentive plans that are specific to the groups within the customer organization. There are a total of 54 groups for which incentive plans are implemented over a medium timeframe. The operational importance of this project to the customer was rather low. To minimize the risk of negatively affecting other more critical projects by modifying common components, it was decided that such interaction should be avoided. That is why there was no reuse, although there would have been opportunities for reuse. The fact that because of the situation there was a conscious decision not to reuse makes this project a special case. The participants in this project have a special perspective that makes them play an important role in the first two steps of the data collection in which the candidate success factors are generated.</td>
</tr>
</tbody>
</table>

factors were considered as possible drivers for variations of project reuse success. Although the literature has presented these factors as drivers for an organizational reuse capability, their ability to change across projects suggests that they may be responsible for project variations.

This initial set of factors was expanded by the set of candidate project-level reuse success factors obtained from the field organization’s software engineers.
who participated in an NGT session (Delbecq, Van de Ven, & Gustafson, 1975). The field company’s vice president (VP) responsible for custom software development had selected eight project leaders, designers, and developers to participate in the NGT session. Based on the VP’s assessment, all participants had sufficient knowledge to understand what factors could contribute to project reuse success. Prior to the session with the actual group, the NGT design was pilot-tested with six information systems professors and doctoral students. Based on the pilot test, the introduction and questions for the idea-generation stage were finalized (see Appendix). The session was held at the company headquarters and took two and one-half hours. It consisted of three stages: idea generation, clarification, and voting (Delbecq et al., 1975). The NGT session was conducted in a professional and unbiased manner by a facilitator (not the researcher) who had successfully run more than a hundred NGT sessions with various groups. In the idea-generation stage, candidate factors affecting project reuse success were collected in a round-robin fashion until every participant had completely exhausted his or her ideas. In the clarification stage, participants were encouraged to raise questions, again in a round-robin fashion, to completely understand every idea in a common way. This was also the stage where ideas were consolidated into a smaller number as certain ideas subsume others, ideas are grouped or merged into others, and so on. It resulted in a list of 37 initial candidate factors. In the final stage of the NGT session, the participants were asked to cast their votes on the six factors they considered most important in terms of their influence on reuse success of software projects. Thus, each participant had the option to either support the
Table 3: Project-level factors from NGT session and literature.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Influence</td>
<td>a. Client’s Budget and Time Constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b. Perceived Value of Reuse by the Client</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c. Client’s Fear of Interconnectivity (threat of multiple failures due to multiple use of a defective component)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Culture</td>
<td>d. Degree of Promotion/Emphasis of Reuse by the Leader/Developing Team During Project Development</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>e. Degree of Training and Incentive of Reuse on the Project</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Attributes</td>
<td>f. Interaction with other Systems (developed within the same reuse framework)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g. Project Sequence (a project is conducive to reuse if a similar project has been done earlier)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>h. Project Domain (degree of suitability of reuse for the particular domain/fit of repository)</td>
<td>+</td>
<td>+</td>
<td></td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer Reuse</td>
<td>i. Experience in Recognizing Reuse Patterns</td>
<td>+</td>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience</td>
<td>j. Understanding of the Company’s Reuse Model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>k. Knowledge of Component Availability/Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Factors he or she initially contributed to the session, or to vote for other participants’ factor nominations. It was apparent that the participants consolidated their votes on a set of 11 factors, which was considered the outcome of the NGT session (Table 3).
Based on the factors’ definitions from the literature and on the definitions that were discussed during the NGT session, three researchers who were familiar with the reuse paradigm classified the 11 factors into four conceptual groups: *client influence*, *project culture*, *project attributes*, and *developer reuse experience*. *Client influence* factors describe the constraints, concerns, or attitudes of the client toward software reuse for a project. *Client* refers to the group of contact persons in the client organization, rather than the client organization itself. *Project culture* reflects the different degrees of emphasis put or incentives offered (tangible or intangible) on reuse by MBA Technologies for various projects. *Project attributes* address issues that are inherent to the project itself, such as the domain and the degree of interaction with other systems. *Developer reuse experience* describes experience and knowledge of the project team members regarding reuse and reusable components. Although classifying the factors into high-level conceptual groups was not necessary at this stage, the grouping presented an effective working framework. By way of follow-up interviews, participants of the NGT study were asked the efficacy of the conceptual grouping. The group’s responses validated the grouping by confirming that the framework helped in enhancing their understanding of the relationships between the various factors.

The outcome of this stage of the research was to propose a set of candidate factors that attempts to explain why project reuse success varies within successful systematic software reuse settings. The participants of the NGT session identified a set of project-level candidate factors. A comparison of this set with the set of literature-identified candidate factors showed that none of the already known factors that can affect reuse on the project level were omitted from the analysis. While the result provided evidence for the quality of the factor list, it also showed that this research was expanding the body of knowledge by complementing the known set of project-level factors with newly obtained reuse drivers.

**Case Analysis and Factor Validation**

The second step of the triangulation was to validate the 11 candidate factors by analyzing case evidence. To do so, the values of the factors needed to be assessed with respect to each of the five projects. The participants in this stage were selected based on their involvement in the projects. For each project, two employees evaluated the factor values in structured interviews. Two people were involved in multiple projects; thus eight participants were selected for the five projects. Seven of these interviewees were also involved in the NGT session. The participants were asked to state the values of the candidate factors with respect to their project(s). The questions are listed in the Appendix (Figure A2). For a small set of questions, consensus was not achieved. The answers to those questions were communicated to the participants and they were given the opportunity to either clarify or reevaluate the assessments. After one iteration of this process, the responses were in agreement. The answers are included in the Appendix (Table A1).

The reuse percentages for each project were calculated for each of the levels of abstraction (low-, medium-, and high-level code) as well as for the entire project, providing 20 data points for the 5 projects (details on the calculation of the reuse rate are presented in Figure 3). To analyze which candidate success factors affect
The reuse rate of the projects are calculated based on a method suggested by Basili et al. (1996) for the assessment of the degree of reuse in object-oriented code. They looked at program reuse in an object-oriented environment as part of a study to assess its impact on quality and productivity in object-oriented systems. The approach has been modified to apply it to a component framework rather than a strictly object-oriented class framework. They describe the size of a system $S$ as a function ($\text{Size}(S)$) with a series of properties. The following is a subset of these:

- Size cannot be negative.
- Size is expected to be zero when a system does not contain any component.
- When components do not have elements in common, size will be additive.

Reuse($S$) is introduced as the amount of reuse in a system $S$; hence it is an instance of the size metric. Therefore the three properties of $\text{Size}(S)$ also apply for Reuse($S$). Reuse($S$) is subject to the following three cases:

1. Component $C$ belongs to the reuse library and has been included in the system $S$ without modifications:
   \[
   \text{Reuse}(C) = \text{Size}(C).
   \]

2. Component $C$ has been created by changing an existing component:
   Reuse can be estimated as
   \[
   \text{Reuse}(C) = (1 - \%\text{Change}) \times \text{Size}(C)
   \]
   Since $\%\text{Change}$ is difficult to obtain, Basili et al. suggest and justify the following simplification:
   - $C$ is more than 25% modified: $\text{Reuse}(C) = 0$
   - $C$ is less than 25% modified: $\text{Reuse}(C) = \text{Size}(C)$.

3. Component $C$ was totally written:
   \[
   \text{Reuse}(C) = 0.
   \]

Given this classification, the reuse rate of a system $S$ is
\[
\text{ReuseRate}(S) = \frac{\sum \text{Reuse}(C)}{\text{Size}(S)} \quad \text{where } C \in \text{Components}(S).
\]

Basili et al. (1996) assume the existence of a size metric. Lines of code (LOC) is used as a size measure. The problems with using LOC as a measure of development effort do not apply if the following assertions hold:

1. *The code is written within the same subject domain.*
   Reuse is measured in a specific business domain. Similarity of the algorithm type ensures that LOC is a good measure for the program complexity, since the average complexity per line of code remains constant across code of the same type.

2. *All components are written in the same programming language.*
   The problem that LOC does not adjust for differences in programming languages does not apply to this company.

3. *The ratio of Comment LOC/Noncomment LOC (CLOC/NLOC) and the amount of white space used is constant across components.*
   The software development company follows a rigid methodology. This suggests consistent programming style, including use of comments and white space. This presumption has been confirmed by sampling the code of the project. Hence, the concern that LOC does not adjust for different programming styles does not apply.

The assertions hold for MBA Technologies. Thus, LOC is a suitable measure for the development effort in this development context and it can be used in the context of the reuse metrics.
Rothenberger 95

project reuse success with respect to the five software projects, the method of pattern analysis was used. The method was presented in a paper on theory building from case studies (Eisenhardt, 1989). It must be stressed that we do not expect statistical validity from pattern analysis. It is rather a qualitative analysis of reuse in each abstraction level of the five projects, as well as the total project reuse. Thus, the results of this step were one part of our triangulation effort; they were subject to further verification through the subsequent analysis steps.

For easier pattern recognition, the projects were sorted by their reuse rate on each level of abstraction, as well as by the project reuse rate. A factor that changed its value in a consistent direction from the project with the highest to the project with the lowest reuse rate was attributed a pattern. In addition, patterns between factor values and reuse percentages on each level of abstraction (low-, medium-, and high-level code), as well as the overall reuse rate of the projects were explored. The three levels of abstraction were examined separately, because not every factor necessarily has an equal effect on the reuse percentage on all structural levels. (Table A1 of the Appendix illustrates how patterns were identified for the total project reuse rate; patterns for the three levels of abstraction were identified accordingly). The results of the pattern analysis for the low-level, medium-level, and high-level code, as well as for the overall reuse rate of the project are presented in Table 4. The checkmarks in the respective columns represent the existence of patterns for a factor. Patterns for the total project or the low-level code, which is the main source of reuse in the company’s development setting, indicate a strong linkage of the factor to reuse success; patterns that only exist for the less important high-level and medium-level code indicate a weaker linkage to reuse success. There was evidence that the client influence factors, project culture, and project attributes appear to strongly affect the success of systematic reuse projects. The analysis also indicated that developer reuse experience has a limited effect on reuse success. Thus, the comparative case studies generally supported the developer perceptions captured by the NGT session.

**Interviews**

As a third step to the triangulation approach, structured interviews with open-ended questions were conducted. The interview approach provided a deeper understanding of how and why the factors affect the success of reuse projects (Yin, 1994), and provided validation of the success factors by obtaining an explanation of the patterns discovered in the previous part of the research. The nature of the patterns found in the cases was explained with the insights obtained in the interviews. The eight participants in the NGT session, as well as the vice president of software development, were questioned in a total of nine interviews, lasting from 25 to 45 minutes each. The interviewees were asked to explain how each reuse factor proposed in the NGT session affected the reuse level of a project. The patterns found in the case studies were not shared with the participants at this time. The interview answers revealed what happens in a software project when reuse is affected by each of the factors (the outline used for the structured interviews is presented in the Appendix). The insights obtained in this stage were analyzed and related to the results of the comparative case studies. Three stories, discussing projects with
Table 4: Results of the cross-case pattern analysis.

<table>
<thead>
<tr>
<th>Conceptual Group</th>
<th>Project-Level Factor (as identified in the NGT Session)</th>
<th>High-level code pattern</th>
<th>Medium-level code pattern</th>
<th>Low-level code pattern</th>
<th>Total project pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Influence</td>
<td>a Client’s Budget and Time Constraints</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>b Perceived Value of Reuse by the Client</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>c Client’s Fear of Interconnectivity (threat of multiple failures due to multiple use of a defective component)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Culture</td>
<td>d Degree of Promotion/Emphasis of Reuse by the Leader/Developing Team During Project Development</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>e Degree of Training and Incentive of Reuse on the Project</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Project Attributes</td>
<td>f Interaction with other Systems (developed within the same reuse framework)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>g Project Sequence (a project is conducive to reuse if a similar project has been done earlier)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>h Project Domain (degree of suitability of reuse for the particular domain/fit of repository)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer Reuse</td>
<td>i Experience in Recognizing Reuse Patterns</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience</td>
<td>j Understanding of the Company’s Reuse Model</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>k Knowledge of Component Availability/Capability</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

different degrees of reuse success, are presented below: one story for the three projects that are in the middle ground, one for a project that is the prototype of reuse activity, and one for the project that achieved no reuse.

The middle-ground projects

Three projects fall into this category: demand planning and invoicing have achieved reuse rates of about 67 percent. Order capture performed slightly lower with 50.5 percent. The case data and the interviews showed that most requirements for successful reuse were met for these three projects. Nevertheless, there were also differences between the projects that explain the variation in reuse success.
Project sequence was important to maximizing reuse and, among other factors, responsible for the reuse rate differences of the projects in the middle-ground category. Ideally, the core system must be completed before peripheral systems are developed. One strategy may be to develop the projects as closely to the actual business cycle as possible (in the context of a supply chain system, beginning with order and ending with invoicing). The development of overlapping concurrent projects must be avoided, since similar components would need to be written separately. Assuming sequential development, such components could contribute to reuse. A project sequence that is ideal for reuse can maximize the reuse opportunities by allowing the use of earlier written components in later projects. The timing of the development of order capture was not conducive to reuse; thus a slightly lower reuse rate than in the other two projects was obtained.

The degree of promotion of reuse by the project leader may also have been responsible for the differences of the projects within this group. Traditionally, promotion and emphasis of reuse has been viewed as an organizational factor that does not vary across projects. However, the interviews have shown that the promotion and emphasis of reuse could change across projects for two reasons:

1. Management and the project leader can adjust the degree of promotion to meet the client’s attitude toward reuse.
2. The level of experience of the project leader can affect the internal promotion of reuse on a project. A more experienced project leader tends to put a higher emphasis on reuse during development. In this case, the reuse experience factors moderate the level of internal promotion of reuse.

In MBA Technologies’ case, promotion and emphasis of reuse has been reduced from the organizational average, because of reason 1.

The demand planning project was the only project with low client’s budget and time constraints. The low client pressure allowed the developers to achieve a high reuse rate of components with medium-level abstraction, which is 84.8 percent—far higher than in any other project. As discussed earlier, the component retrieval of medium-level components was less effective than the highly automated retrieval of low-level components. Tight budget and time constraints would have prevented programmers from spending the time necessary to find reusable components with such a labor-intensive retrieval processes. Thus, reuse opportunities would have been missed. The absence of this type of client pressure allowed the developers to obtain the full reuse potential from the medium-level components in this project. Nevertheless, because of the low share of medium-level code to the overall project, the reuse success in this category did not make a difference for the overall reuse success of this project.

The three developer reuse experience factors only affected reuse of high-level components. As mentioned earlier, the retrieval of this type of components relied strongly on the developers, as it is not automated in the MBA Technologies environment.

Because of the low share of high-level components to overall project reuse in the case environment, the differences in the developer reuse experience factors affected total reuse in the MBA Technologies projects very little. Nevertheless, the
Project-Level Reuse Factors

effect of such factors may be important in other, nonautomated reuse environments. The following discussion applies mostly to organizational reuse settings that rely on developer skills for component retrieval, rather than on an automated process. Accordingly, the experience in recognizing reuse patterns enables designers and developers to recognize places in software projects where reuse is likely to happen. The understanding of the company’s reuse model is specific to the model of an organization and ensures that the right parts of the model will be reused. It is a long process for developers to acquire such understanding because it can only partially be obtained through training. Structured paradigms learned in training or on previous projects do not include systematic reuse procedures and need to be unlearned, which can only be achieved through work on reuse projects. Finally, the knowledge of component availability and capability helps to retrieve components and lowers the likelihood to miss components, avoiding unnecessary redevelopment. The key to understanding a component’s capabilities is actually using the component; hence developers who have worked on multiple projects will have good knowledge of the repository’s contents. Interviewees stated that the methodology helped to identify components only to a limited extent. An ideal cataloging process and tool set would reduce the relevance of component knowledge for the reuse rate achieved in software projects. One respondent shared his view on the relevance of experience:

“[The] questions [of designer and developer experience] hopefully will be de-emphasized over time. . . . What they point out is a lack of an underlying tool that in our vision should be very intuitive and should not require a lot of extra training, it should not require a lot of knowledge. [Without such tools] you build up an awful lot of knowledge of where things are, what components are, how they work, and how they can be best used, and people are walking around with that in their head. . . . The most important valuable information is that proprietary knowledge of. . . . how to reuse things and how to extend them.”

In summary, the three projects obtained a satisfactory reuse rate, because of most project-level factors being conducive to reuse. Differences between the projects were explained by the project sequence and variations in the promotion of reuse. There is also evidence that in less automated reuse environments, the client’s budget and time constraints and the developer reuse experience may more strongly affect reuse success than they did in the context of the case environment.

The prototype of reuse activity

The Trials project achieved the highest overall reuse rate, 76 percent. While the factors that helped the middle-ground projects to achieve their reuse rates were also conducive to reuse in this project, one additional factor stood out because its value differed from the remaining projects. The perceived value of reuse by the client was highest in the Trials project. It must be stressed that none of the projects scored “high” in this category. Nevertheless, Trials had a medium perceived value of reuse, while the other projects were assessed as low. The perception of reuse by the client depends mainly on the effort that is invested up front in education about reuse benefits. At MBA Technologies, clients were usually educated about reuse at the beginning of a project to get them to agree to use the reuse paradigm. However, during later stages of a project, their support of reuse often declined. In those instances, clients communicated their low perception by not allowing the
needed planning time for reuse. One interviewee explained what can happen when the perceived value of reuse is low:

“You are actually designing for reuse and it takes a lot of up-front design time. . . . I think the magic number is six months—if you go for six months or more without having shown anything, the client starts to get antsy and he might say, ‘just start coding something.’”

In the Trials project, the client gave the developers the time needed to plan for reuse. Thus, the client influence factors made the difference between a middle-ground and a high-reuse project. Although the project had most of the other items in place, without the client support of reuse the project would not have excelled to this degree.

**The project with low reuse**

The Incentives project obtained no reuse. It seemed unusual that in an otherwise successful reuse environment, a project was developed 100 percent from scratch. This section will explain what drivers led to its development without reuse. The project was in the domain of decision support and performance measurement and was therefore different from the previously developed operation- and transaction-driven systems. Thus, the project had a low reuse potential; fewer desired components were in the company’s repository. The case analysis has shown that the domain particularly affected reuse achieved from the more domain-specific high-level code. Since MBA Technologies achieved little reuse through high-level code, the domain’s effect must have been low. That means that the zero reuse rate of the Incentives project cannot solely be explained by the domain mismatch, as there must have been reuse opportunities in the non-domain-specific low-level components.

Two other factors explained why low-level reuse was not achieved; the interaction with other systems was limited to read-only access to common operational data. It was the only project that scored low in this category. According to the respondents, low interaction decreases the number of reusable design patterns and components to choose from. Designers and developers are less inclined to look for reuse opportunities where there is low interaction between systems. The operational importance of this project to the customer was rather low and therefore the client felt that it should not affect any of the other more critical projects. This fear of interconnectivity was triggered by previous negative experience. In the past, MBA Technologies had to shut down two systems at the client’s site to fix an error in a frequently used module. Such fear is partially justified: when components are reused, often, modifications are made and complexity is added. These changes can affect other projects that are using the same component. However, knowledge of the dependencies among projects may prevent the team from making modifications that do not work with previous projects. The likelihood of such failure increases as complexity grows through multiple reuse of the module. A critical component that had led to failures is used in a majority of MBA Technologies’ projects. To prevent future problems, it was split into multiple components with lower complexity. One employee described what software developers need to be aware of:
“[When you are adding functionality to a component], complexity is building up and at some point there might be a critical mass. . . . When it does fail it is more catastrophic than a small modularized component . . . Each implementation product or company has to recognize where the critical mass needs to be diffused and what size to keep your base components.”

Because of the fear of interconnectivity the client elected to avoid reuse in the Incentives project, as its operational importance was perceived as low, thus not justifying the risk of affecting other crucial applications. The use of multiple versions of crucial components could have lowered the risk; however, MBA Technologies was not able to alleviate the client’s fear in this instance.

In summary, no domain-specific reuse was achieved, because of differences between the project domain and the repository. Lower-level reuse was not achieved, because of the lack of interaction with other systems and the client’s fear that existing software may be adversely affected.

LIMITATIONS AND FUTURE RESEARCH

In an ideal world, the participants in the triangulation stages of this research would have been three different groups. Because of the size of the organization, overlap between these three groups could not be avoided. When designing the study, careful attention was paid to this potential shortcoming. The research was designed in a way that allowed for a minimal effect of the participant overlap across the stages of the research on the results of the study. Initially, the organization had no formal reuse measurement program in place. Thus, the reuse success of the individual projects was unknown to the employees. The actual reuse rates were not communicated to the employees until the conclusion of the study. Thus, participants of the NGT session were not guided by individual project successes when they identified the candidate factors. Further, the project values of the second stage were not biased by respondents’ knowledge of project success. The interviewees in the third stage did not have the reuse success information necessary to confirm their understanding of the factor effects; thus there is reason to believe that the overlap with the second stage did not affect the triangulation approach, either. Tapping different aspects of the participants’ experiences also helped to minimize the effects of the overlap. The NGT session aimed at intuitively identifying the project reuse success factors; the interview stage aimed at exploring how the proposed project-level factors may affect reuse success. Although there was participant overlap in both stages, consistency of the intuitive factor identification and the mechanics behind the factors’ affects on reuse indicate that the candidate factors have merit.

External validity deals with the question of whether the findings are generalizable to reuse settings beyond the organization studied. External validity has been a major barrier for case research (Yin, 1994). Critics argue that the small sample size inhibits the researcher’s ability to generalize. Thus, it must be stressed that the goal of case study research is not to obtain statistical generalizability. Nevertheless, the results of a case study can be generalized to a broader theory. Such generalization is not automatic (Yin, 1994). While the organization in this research meets the requirements typical for software developers employing reuse, the possibility that variations in the development methodology may affect the nature of project reuse success factors cannot be excluded. Only replications of this study
in the context of other software reuse settings can provide the ultimate proof of external validity. This is even more important to understand the applicability of the factors to settings that substantially differ from the case organization in process maturity or reuse capability. In addition, larger-scale surveys that require software developers to assess the value of the project reuse success factors as well as the degree of reuse success of individual software reuse projects may further enhance our understanding of the applicability of the model to other settings. Our future research shall be aimed in these directions.

The review of existing organizational factors has shown that the sets of factors identified are only partially overlapping. There is an opportunity to explore why these researchers obtain such different results. One possible explanation may be a domain dependence of organizational factors. Eventually, a unified theory of the effects of organizational and project-level factors on reuse success may be developed. Such research may also expand the definition of reuse success beyond the reuse rate to include cost and quality in a success measure. The availability of such a measure may also improve organizational decision making on reuse methods.

SUMMARY AND CONCLUSION

Identifying the factors that affect reuse success in a systematic reuse development environment allows 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. The outcome of this research is a set of factors that explain why project reuse success varies in an overall successful systematic software reuse setting.

Contributions

The contributions of this study are threefold. First, the research has reexamined literature-identified reuse success factors to assess whether they can be responsible for reuse-level changes across projects in a given development environment. Factors in the areas of project culture, project attributes, and developer reuse experience have been found to be responsible for project-level changes in reuse success.

Second, factors that were entirely unexplored by the literature have been developed and their effect on project-level reuse success has been explained. For example, the notion of client influence factors has been entirely unexplored by the existing literature. The research also added to the existing groups of project attributes and developer reuse experience by finding additional factors that affect reuse projects.

Finally, this research expanded our understanding of how the existing and newly identified factors affect reuse on a project level. The interviews have provided insights on the importance of the factors and their effect on various aspects of the development process with reuse.

Managerial Caveats

Many organizations can report great reuse success stories for a few projects. However, most organizations find it difficult to replicate this success and attain a
consistent level of reuse. Such scenarios suggest that the organization has a high reuse capability as established by the organizational success factors. The problem that some projects achieve considerable reuse while others don’t may be explained by the existence of project-level success factors that have been introduced in this research. Variations between the project-level success factor values of subsequent projects can make the difference between reuse success and reuse failure. Based on this study, organizations should not only try to improve organizational factors, but also the newly introduced project-level reuse factors. This will allow the development group to ensure that the software projects can live up to the development environment’s reuse capability.

The project-level factors proposed in this study were obtained in an organizational environment that was conducive to reuse. In other development environments, the effect of project-level factors on reuse may be limited. It will need careful attention in future research and practice to determine to what extent organizational factors that are not in support of reuse can be overcome on the project level. The results of the study suggest that in organizations with a high reuse potential, a project will achieve high reuse if:

- The client perceives the value of reuse to be high with respect to the project.
- The client views interconnectivity as an opportunity, rather than a threat of failure.
- The software development group promotes reuse heavily during the development of the project.
- The system is tied in with other software that has been developed using the same reuse paradigm.
- The sequence of projects allows new projects to benefit from components written for previous projects.

A second set of factors exists that was, to the case organization, less critical than the previous one. The study indicated that these factors affect reuse, especially if the component retrieval process is not automated. That means that the developers relied on personal knowledge of the components to find suitable modules in the repository. In such an environment, reuse will improve if:

- The project is in the same domain as previously written projects.
- Designers and developers on the project have several years of experience in recognizing reuse patterns.
- Designers and developers have a good understanding of the reuse model employed and know the contents of the repository and the capabilities of the reusable components.

Not all of these factors can be manipulated by the organization. For example, if the project is the first of its domain, there is little to be done to change this. However, knowledge of the effect this factor may have on reuse will allow the development team to better assess development effort needed to complete the project. Other factors may be manipulated, such as the degree of the promotion of reuse on the project team or the sequence of the projects. Often, communication with the customer will be necessary to establish an environment that is conducive to reuse. This might allow both parties to agree on a suitable project sequence, as well as
to establish trust in reuse as a means to improve the software process and the final product. [Received: October 29, 2001. Accepted: January 2, 2003.]

REFERENCES


**APPENDIX**

**Questions for the NGT Session and the Structured Interviews**

**Figure A1:** NGT session question.

For the purpose of this meeting, we make the following assumptions about “reuse”:

- We are only looking at code that has been hand-coded (reused or nonreused), such as `<company terminology>`, `<company terminology>`, and `<company terminology>`. We are disregarding the generated part of each project.

- Reuse is defined as code that has been written for an earlier project and that has been reused in the current project without significant modifications (<25% of development effort of the component). We are not counting the repeated use of the same component within a project as reuse.

- Among the projects we looked at, there is a difference in the levels of reuse.

Judging from your experience with all projects you have worked on at MBA Technologies, what do you think contributes to the variations in reuse levels (reuse rates) across projects?

Please write your ideas in bullet points below:
Figure A2: Questions for the assessment of the factor values.

The answers to the following questions are marked on a three-point scale: low, medium, high.

1. The client’s budget and time constraints for the project were
2. The client perceived the value of reuse for the project to be
3. The client’s fear of interconnectivity due to failure of the reused component with respect to the project was
4. During the project, MBA Technologies’ internal promotion of reuse was
5. The experience of the developers and designers working on the project in recognizing reuse patterns was
6. The degree of understanding of MBA Technologies’ reuse model by the designers and developers working on the project was
7. The degree of knowledge of component availability and capability by the designers and developers working on the project was
8. The degree of reuse training and incentive for developers on the project was
9. The interaction with other systems (user and MBA Technologies systems) of the project was
10. To what extent was the project sequence conducive to reuse?
11. To what extent was the project domain conducive to reuse?

Figure A3: Structured interview outline.

1. The client’s budget and time constraints
2. The client’s perceived value of reuse for the project
3. The client’s fear of interconnectivity due to failure of a reuse component
4. The degree of MBA Technologies’ internal promotion of reuse during a project
5. The level of experience of the developers and designers working on the project in recognizing reuse patterns
6. The degree of understanding of MBA Technologies’ reuse model by the designers and developers working on the project
7. The degree of knowledge of component availability and capability by the designers and developers working on the project
8. The degree of reuse training and incentive for developers on the project
9. The degree of interaction with other systems
10. The project sequence
11. The project domain
Analysis Supplement

Table A1: Factor values and patterns for the project reuse rate.

<table>
<thead>
<tr>
<th>Project</th>
<th>Reuse Rate</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
<th>(g)</th>
<th>(h)</th>
<th>(i)</th>
<th>(j)</th>
<th>(k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trials</td>
<td>76.0%</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Demand Planning</td>
<td>67.6%</td>
<td>L</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>M</td>
<td>M</td>
<td>L</td>
</tr>
<tr>
<td>Invoicing</td>
<td>67.4%</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>M</td>
<td>L</td>
<td>M</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>Order Capture</td>
<td>50.5%</td>
<td>H</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Incentives</td>
<td>0.0%</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>L</td>
<td>N/A</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

- L = low; M = medium; H = high
- Letters (a) through (k) are referencing project-level factors as listed in Table 4.
- Patterns exist in columns b, c, f, and g (i.e., factor values change in a consistent direction from the project with the highest to the project with the lowest reuse rate).

Marcus A. Rothenberger is assistant professor at the University of Wisconsin–Milwaukee. He received a PhD in information systems and an MBA, both from Arizona State University. He holds an undergraduate degree in computer science and business from the Technical University of Darmstadt, Germany. Previously, he worked for Deutsche Bank AG in the information engineering area. His current research interests include software reusability, performance measurement, software engineering, and enterprise-system adoption. He is a member of the Association for Information Systems (AIS), the Decision Sciences Institute, and the Association for Computing Machinery (ACM).