Ask the Engineers: Exploring Repertory Grids and Personal Constructs for Software Data Analysis

Lucas Layman[*], Carolyn Seaman[†*], Davide Falessi[*], Madeline Diep[*]

[*]Software and Process Analytics
Fraunhofer Center for Experimental Software Engineering
College Park, MD, USA
{llayman, dfalessi, mdiep}@fc-md.umd.edu

[†]Dept. of Information Systems
University of Maryland, Baltimore County
Baltimore, MD, USA
cseaman@umbc.edu

***Abstract*—Maturity in software projects is often equated with data-driven predictability. However, data collection is expensive and measuring all variables that may correlate with project outcome is neither practical nor feasible. In contrast, a project engineer can identify a handful of factors that he or she believes influence the success of a project. The challenge is to quantify engineers' insights in a way that is useful for data analysis. In this exploratory study, we investigate the repertory grid technique for this purpose.**

The repertory grid technique is an interview-based procedure for eliciting "constructs" (e.g., adhering to coding standards) that individuals believe influence a worldly phenomenon (e.g., what makes a high-quality software project) by comparing example elements from their past (e.g., projects they have worked on). We investigate the relationship between objective metrics of project performance and repertory grid constructs elicited from eight software engineers. Our results show correlations between the engineers' subjective constructs and the objective project outcome measures. This suggests that repertory grids may be of benefit in developing models of project outcomes, particularly when project data is limited.

***Index Terms*—repertory grids; practitioners; software data analytics**

I. Introduction

Software engineers have long sought to quantify factors that measure, correlate, and predict project outcomes. Data collection can be costly and is often an overhead expense on projects, thus, organizations must focus their data collection efforts on a few, hopefully impactful, factors. In practice, selecting which data to collect is difficult for two reasons:

1. There are hundreds of factors that potentially impact project outcome [1]–[3];
2. When asking which factors are most relevant to a project, the answer is usually "it depends" [4], [5].

In lieu of a costly, comprehensive data collection program, where should project managers begin their data collection process? One logical starting point is to consult with the software engineers on the projects, who presumably have intimate knowledge of the underlying factors that impact the success of their projects. The challenge is to extract and quantify these factors from individual engineers, from the ground up, such that it can be used in lieu of or to augment data-driven software modeling.

This study investigates the feasibility of repertory grids [6] as a technique for eliciting meaningful factors from developers that they believe impact quality and productivity on their software projects. The repertory grid technique originated in the field of psychology and is used to elicit personally-relevant "constructs" (e.g., complexity, level of customer interaction) related to a topic (e.g., quality, productivity) based on comparing example "elements" (e.g., projects the interviewee has worked on), which are then rated on a scale for each construct. Our long term research goal is to determine: a) if personal construct ratings can be reliably used to predict project outcomes; and b) if personal construct ratings can be used to augment objective project and process metrics to produce superior prediction and classification models.

We conducted repertory grid interviews with eight software engineers at a web application development company in the USA. We analyzed the repertory grids to determine how well the elicited constructs correlate with project outcomes captured by quantitative measures of quality and productivity. In particular, we evaluate the following research questions:

RQ-1. Which repertory grid constructs relate to project quality and productivity outcome measures?

RQ-2. Are some engineers better able to identify constructs relevant to productivity or quality that others?

RQ-3. Are individuals' quality and productivity constructs reflected in the objective project metrics?

We discuss repertory grids in section 2, describe the study context in section 3, and provide analysis in section 4.

II. Repertory Grid Background

The repertory grid technique originated in Personal Construct Theory [6] and is an interview-based procedure for eliciting a person's "constructs" that form their view of some phenomenon in the world. For example, one can ask a developer, "what do you think makes a project successful?" They may reply with the constructs "accurate scheduling, experienced developers, and a responsive customer". A central tenet of Personal Construct Theory is that people form their constructs by looking at the contrasts between examples [7]. The repertory grid interview systematically elicits constructs from individuals and starts by defining a topic, e.g., "factors that impact quality of a software project". The following steps are then iterated to create a set of constructs:

1. Agree on a set of elements to compare, e.g., a list of software projects familiar to the interviewee.
2. Randomly select three elements.
3. Ask the interviewee to describe in what way two elements are similar but different from the third with respect to the topic. The response forms the construct. Write down how the two are similar on the left side of the construct row, and how the third is different on the right side (the poles). Discuss the construct until it is well-defined.
4. On a scale (usually from 1-5), have the interviewee rate the entire set of elements, with 1 corresponding to the left pole of the construct, and 5 corresponding to the right pole.
5. Repeat from Step 2 until no more constructs are generated, or until the interview time limit is reached.

Manual content analysis and quantitative techniques are applied to the resulting grid to gain insight into the constructs of individuals, common constructs from multiple individuals, and perceptions of multiple individuals on shared elements. An example of a completed grid is shown in Fig. 1. For an in-depth discussion of repertory grid interviews and analyses, see Jankowicz [7]. A survey of studies using repertory grid analysis in empirical software engineering can be found in Edwards [8]. To our knowledge, this work represents the first use of repertory grids to elicit data to be analyzed with respect to traditional project outcome measures, such as defect density.

	ProjA	ProjB	ProjC	ProjD	ProjE	ProjF	ProjG	
Divergent stakeholders	5	4	1	1	1	3	3	Cohesive stakeholders
Incomplete/unclear requirements	5	4	2	1	3	4	4	Clearly-defined requirements
Complicated business requirements	4	3	4	2	1	4	5	Streamlined business requirements
Unfamiliar tech	5	5	2	3	4	4	4	Familiar tech
COTS-based	5	5	1	1	4	4	3	Custom development
Insular customer	5	4	3	1	1	4	4	Collaborative customer

Fig. 1. Example repertory grid

III. Methodology

The goal of this exploratory study is to assess the potential of repertory grids for generating important, otherwise uncaptured variables (i.e., constructs) that relate to project quality and productivity. We first elicit repertory grids from a sample of software engineers, and then compare the construct ratings to the actual project outcomes.

A. Interviews

Repertory grid interviews were conducted with eight software engineers at a web application development company in the USA. The interviewees fell into three categories: three project managers, four developers, and one requirements analyst. The interviewees were selected from a convenience sample of engineers who had worked on multiple projects at the company. The interviews were conducted individually - two were conducted face to face, and six by videoconference. The lead author conducted each interview with other authors present for half the interviews. Interview sessions were limited to 60 minutes. The repertory grid was created in an Excel spreadsheet that was edited by the interviewer and viewed by the interviewee in real time. The topic for the interview was "factors that impact quality and productivity on software projects". The interviewer reminded the interviewees of the topic prior to eliciting each construct. The elements (projects) in the grids were a set of software projects for which the company had already collected metrics. Each interviewee was asked if he/she had worked on these elements (projects), and if not, those elements were removed and the interviewee was asked to provide the names of other projects on which he/she had worked. The interview procedure then followed the steps enumerated in Section 2.

B. Project Outcome Measures

The development company tracks various metrics for each released version of their projects in a JIRA project management system, including lines of code touched, code complexity metrics, number of team members, time spent, tasks completed, and post-release defects reported. We defined project quality and productivity outcome measures (i.e., dependent variables) derived from these data:

Defect density – number of reported production defects per LOC touched (added + deleted + modified LOC).

% severe – percentage of released versions containing a severe defect. In practice, the development company focuses more on preventing severe defects from making it to production rather than on overall defect density.

Efficiency – the average number of LOC touched per implementation task, which includes both change requests and functional enhancements. Lower values indicate higher efficiency. The implementation tasks must have similar size for this outcome measure to be meaningful. The company is rated at CMMI Level 5, and the development process requires uniformity in the definition of implementation tasks.

The unit of analysis in the repertory grid interviews was an entire project, but the project metrics were captured per project per version. For this study, the mean defect density and mean efficiency were calculated across all versions of a project to generate outcome measures at the project level.

IV. Results

A total of 46 constructs were elicited from 8 participants, based on their experience on 22 different projects (elements). Seven of the participants provided 5-6 constructs, while one provided 9. The number of elements in each grid ranged from 4 to 8 with a median of 7. Approximately 1/3 of the elements in each grid referred to projects for which there were no recorded outcome measures (and so could not be included in the following analysis), but all grids had at least a third of their elements included in the analysis.

A. RQ-1: Which repertory grid constructs relate to project quality and productivity outcome measures?

In total, the participants generated 46 constructs. To answer RQ1, Kendall's τ (preferred over Spearman's ρ for large proportions of tied ranks) using a 95% confidence interval was calculated for each construct's ratings and each outcome measure for the rated projects. The projects were rated on a scale from 1 to 5 according to the poles each construct. For example for a construct for "complexity of customer interaction", 1=low complexity and 5=high complexity.

Of the 46 constructs, 17 were excluded from the correlation analysis because they rated fewer than four projects for which we had objective outcome measures. Thus 29*3 correlations were computed. In our analysis, we consider correlations where $|\tau| \geq 0.65$, which indicated a large effect. None of the correlations were statistically significant at the 0.05 significance level after applying the Bonferroni correction, which is to be expected given the small samples and large number of hypotheses. Thus, the relations below are only examined with an anecdotal, exploratory lens.

Only 6 of 87 (7%) of comparisons had $|\tau| \geq 0.65$. None of the constructs related to the *% severe* outcome measure, and two constructs were correlated with the *defect density* measure. The remaining correlations were four constructs correlated with the *efficiency* measure.

The descriptions of the constructs are summarized from participant discussion and verified with the participants. The two constructs correlated with *defect density* are:

C1. When "done" is not clearly defined by the customer, frequent change requests can cause release delay. (τ=0.72)

C2. Complicated business rules or complicated workflows yield highly-interrelated pieces of functionality, thus making it more difficult to implement and change code. (τ=0.74)

These constructs reflect a blend of requirements complexity and customer communication issues. The net effect of C1 is an increased number of change requests, which research has shown to be associated with increases in the number of system defects [9], [10]. Construct C2 captures the traditional notion of requirements complexity while suggesting that complex requirements lead to a complex system where changes are more challenging to implement correctly.

Four of the constructs significantly correlated with *efficiency*, and these had to do with process execution, architecture, and also requirements:

C3. Compliance with a reference architecture enables developers who are new to the project to better understand project design and become effective more quickly. (τ=0.79)

C4. Following a reference architecture and using familiar technologies promotes efficiency. (τ=0.74)

C5. Poor technical design and coding practice lead to more defects and higher effort required to understand and fix the code. (τ = 0.69)

C6. Adherence to CMMI Level 5 practices promotes efficiency. (τ=0.78)

The reference architecture referred to in C3 and C4 is available to developers in the company as reference point for implementing common business requirements. C5 and C6 suggest that mature practices reduce the cost of change.

B. RQ-2: Are some engineers better able to identify constructs relevant to productivity or quality that others?

In RQ2, we want to understand who we should listen to when building quality and productivity models. The significant constructs came from 3 of 8 interviewees, meaning that three participants did not produce a construct that was significantly correlated with an outcome measure. No single construct was significantly correlated with multiple outcome measures.

Only one participant identified correlated constructs both for *efficiency* and for *defect density*: the Process Improvement Director for the organization. The other participants who whose construct ratings correlated with actual outcomes were the User Experience Director and a Senior Software Engineer.

C. RQ-3: Are individuals' quality and productivity constructs reflected in the objective project metrics?

The web application development company has been certified at CMMI Level 5 and has a mature, metrics-driven approach to project management. Numerous project and process metrics are captured in the company's JIRA project management system, however, we observed that *none* of these constructs are directly captured in those or other metrics. Construct C1 has a surrogate metric captured in JIRA, the number of change requests. The company also uses tools to automatically check compliance with the reference architecture and coding practices in referred to in C3-C5. The company currently uses change request metrics and code quality metrics as inputs to their defect prediction models.

V. Threats to Validity

One challenge is to ensure that the documented constructs represent the interviewee's actual perceptions. Two researchers conducted each interview: the primary interviewer completed the repertory grid with the interviewee while a scribe took additional notes on the interviewee's responses. All authors reviewed the construct summaries (e.g., C1-C6) for accuracy in representing the detailed descriptions of constructs in the repertory grids.

The project outcome measures used in the correlation analyses were averages over the entire lifetime of the project. However, it is possible that the interviewees were not familiar with the project for its entire lifetime, or their constructs were dominated by recent, first, or memorable experiences on the project, rather than "averaged" over the entire project lifetime.

The sample sizes for many of the correlations were very small, i.e., between 4-6 pairs of items, and no statistical significance is implied. Thus, the correlations with large effect size ($\tau > 0.65$) may be due random chance. Certainly, the low number of high-effect correlations also point to this possibility. We emphasize that this is exploratory work on the use of repertory grids for eliciting measures that may correlate with traditional project outcomes – these limitations point to much future work, described in the next section.

VI. Discussion and Future work

The goal of this exploratory study is to investigate the potential of repertory grids as a means to elicit and quantify complex factors that impact project outcome from software engineers. Our hope is that repertory grids can provide a developer-based starting point for a useful software data collection program. Furthermore, quantifications of developers' intuitions and insights may augment traditional metrics to produce superior predictive models.

Using the repertory grid technique, we elicited 46 constructs and two were correlated with project outcome measures for quality, and four for productivity. Ideally, most of the constructs would be correlated with at least one outcome measure. We posit several reasons for this:

1. Software developers have difficulty articulating their thoughts in the context of a repertory grid interview.
2. Software developers are misguided about what factors lead to favorable development outcomes.
3. Project impact factors and their relationship to outcomes are extremely localized, that is, there are no general lessons to be learned.
4. The outcome measures, and their underlying project metrics, are not reflective of developers' definitions of project quality and productivity.

In light of the observations and limitations uncovered in this exploratory study, we will address the following research challenges in future work:

Challenge 1: Defining meaningful outcome measures for quality and efficiency. We believe it is likely that the outcome measures defined in this study, based on available project metrics, are not concordant with the participants' notions of quality and productivity. Can repertory grids be used to define better measures of project quality and efficiency from the perspective of both developers and customers?

Challenge 2: Correcting for interviewee inaccuracy and bias. Some project personnel have constructs that more accurately reflect notions of quality or efficiency for specific projects. How do we identify these "more accurate" persons and their constructs and leverage their insights?

Challenge 3: Combining repertory grid data from multiple interviewees. Constructs from multiple persons who are expert in different facets of a project may provide a more complete and accurate picture of a project. However, the challenge of aggregating constructs and scores across multiple grids is a known problem in repertory grid literature [7]. How do we meaningfully aggregate constructs elicited from different participants, and how do we weight constructs from different sources to obtain accurate insight into project performance?

Challenge 4: Combining repertory grid data with other project metrics. Researchers and practitioners have successfully created predictive models in software engineering using project metrics. But can we do better if we augment existing data with repertory grids, which capture complex factors may be unobservable with traditional metrics? Repertory grids (and other subjective methods) may be particularly useful when there is little or no preexisting project data to model. How do we identify where metrics-based models perform poorly, and how do we integrate models of repertory grid data to improve overall model performance?

Repertory grids are an intuitive, human-oriented method for extracting factors that impact project outcome. The practical challenge is to determine how to effectively use these insights for supporting project management and prediction.

Acknowledgment

The authors thank the web development company and its engineers for participating this study, and Michele Shaw from Fraunhofer CESE for enabling this connection. This work is sponsored by National Science Foundation grant #1302169.

References

[1] K. Petersen and C. Wohlin, "Context in industrial software engineering research," in *3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09)*, 2009, pp. 401–404.

[2] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Inf. Softw. Technol.*, vol. 54, no. 5, pp. 433–447, May 2012.

[3] C. Jones, *Applied Software Measurement: Assuring Productivity and Quality*. New York, NY: McGraw-Hill, 1996.

[4] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann, "Local versus Global Lessons for Defect Prediction and Effort Estimation," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 822–834, Jun. 2013.

[5] T. Dybå, D. I. K. Sjøberg, and D. S. Cruzes, "What works for whom, where, when, and why?," in *Proc. of the ACM-IEEE Int'l Symp. on Empirical Software Engineering and Measurement (ESEM '12)*, 2012, pp. 19–28.

[6] G. Kelly, *The Psychology of Personal Constructs*. New York, NY: Norton, 1955.

[7] D. Jankowicz, *The Easy Guide to Repertory Grids*. West Sussex, England: John Wiley & Sons Ltd., 2004.

[8] H. M. Edwards, S. McDonald, and S. Michelle Young, "The repertory grid technique: Its place in empirical software engineering research," *Inf. Softw. Technol.*, vol. 51, no. 4, pp. 785–798, Apr. 2009.

[9] T. Javed, M. e Maqsood, and Q. S. Durrani, "A study to investigate the impact of requirements instability on software defects," *ACM SIGSOFT Softw. Eng. Notes*, vol. 29, no. 3, p. 1, May 2004.

[10] Y. K. Malaiya and J. Denton, "Requirements volatility and defect density," in *Proceedings 10th International Symposium on Software Reliability Engineering (ISSRE '99)*, 1999, pp. 285–294.