Identifying Essential Competencies of Software Engineers

Richard T. Turley
Colorado Memory Systems Inc.
800 S. Taft Ave.
Loveland, CO 80537
RICKTURL.COMEMSYS@CMS.gr.hp.com

James M. Bieman*
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
bieman@cs.colostate.edu

Abstract

The knowledge and skills of software engineers are perhaps the most important factors in determining the success of software development. Thus, we seek to identify the professional competencies that are most essential. In the first phase of our research, we use the Critical Incident Interview technique to identify essential competencies. The Critical Incident Interview technique is a rigorous method for determining critical job requirements from structured interviews with workers. We use this technique in an in-depth review of 20 professional software engineers employed by a major computer firm. Our review includes an evaluation of biographical and Critical Incidence Interview data for 10 exceptional and 10 non-exceptional subjects. We also analyze competencies identified by software managers. We identify 38 essential competencies of software engineers. Differences between exceptional and non-exceptional subjects were not expected in this first phase of our research. We studied exceptional and non-exceptional engineers to ensure that all competencies are uncovered.

Subject areas: software engineering, large software development, software teams, knowledge and skills of software engineers, software productivity, software psychology

1 Introduction

Much effort has been placed in the development of *engineering* approaches to software development such as software tools, coding practices, and test technology. But the overwhelming determiner of software productivity and quality is still personnel and team capability. Boehm found personnel and team capability to be twice as important as the next most important productivity factor [Boe81]. By studying exceptional programmers, the individual capabilities that most influence performance can be identified [Cur81].

Boehm also cites a 25-to-1 ratio between the most productive and least productive software developers and a 10-to-1 difference in their error rates [Boe88]. Brooks suggests the "use of great designers" as one of five promising approaches to improve software development productivity [Bro87]. One of Boehm's seven basic principles of software engineering is to use "better and fewer people" [Boe83].

Our aim is to determine the attributes that are necessary for exceptional performance, so that the performance of all software engineers can be improved. We report the results from the first phase of a two phase study designed to determine the essential competencies of professional software engineers. In Phase 1 we identify these competencies via the Critical Incident Interview technique. In Phase 2 (to be discussed in a later paper), we perform a quantitative study to differentially relate these competencies to engineer performance.

This study is based on the premise that *exceptional* software engineers exhibit different skills which they apply to the problems of software engineering. These unique skills can be identified by careful study of experienced software engineers.

Our overall goal is to identify the skills, techniques, and attributes that are used by skilled programmers, but not used by less skilled programmers. In this paper, we report the results of our efforts to identify critical professional competencies through in-depth interviews of a small sample of exceptional and non-exceptional software engineers. We evaluate the subjects with a Biographical Questionnaire, and we conduct Critical Incident Interviews of the subjects. In a follow-up study, we will use the identified competencies, larger samples, and objective survey instruments to identify significant differences between exceptional and non-exceptional software engineers.

2 Subjects

Subjects are drawn from five commercial research and development laboratories at three different sites. The subjects develop applications in test and measurement, embedded firmware, and computer aided design.

We use two matched subject pools with 10 subjects in each of the exceptional and non-exceptional pools. The subjects are matched by *time in current organization*. Thus, if an

*J. Bieman's research is partially supported by the NASA Langley Research Center, Colorado Advanced Software Institute (CASI), Computer Technology Associates (CTA) and Storage Technology Inc. CASI is sponsored in part by the Colorado Advanced Technology Institute (CATI), an agency of the state of Colorado. CATI promotes advanced technology teaching and research at universities in Colorado for the purpose of economic development.

Population Summary	Total
#Engineers	252
#SW Engineers	150
#Study Participants	20
# of Exceptional SW Engineers Studied	10
% of Total SW Engineers Deemed Exceptional	6.7%

Table 1: Population Summary

(n=20)

Years at Company in Software	Mean	Std Dev	Range
Exceptional	9.05	3.59	4–15
Non-Exceptional	5.00	1.75	2–7.5

Table 2: Years at Company in Software — Differential

exceptional engineer with four years in the current organization is identified, a second *non*-exceptional engineer with four years experience in the same organization is added to the study. This approach controls for the effect of the organization on the individual's performance. The study does not attempt to control any other factors, since all are possible contributors to exceptional performance.

All subjects are professional software development engineers from a major US corporation (referred to as *The Company* for proprietary reasons) with a minimum of two years of experience in developing software. Each subject has successfully completed a project released to the end user. Table 1 summarizes the population from which the study participants are drawn. The *#Engineers* represents the number of engineers of all disciplines in the total population, while the *#SW Engineers* represents the number of software engineers in the total. The *#Study Participants* indicates the number of engineers that were selected for the study. The *% of Total SW Engineers Deemed Exceptional* is the ratio of the number of exceptional software engineers studied to the total number of software engineers in the population. The population represents a sample of organizational units in The Company.

Subjects are selected by a process in which managers identify the top performers in their organization. Managers were asked to identify an exceptional (top 5% of the organization) and average performing pair of individuals. The pair should have spent the same amount of time in the organization. As a result of this process, manager bias is an inherent part of the research design. Exceptional software engineers are those identified as exceptional by managers. Vessey also used manager assessment as a method (the *"ex ante"* method) for identifying experts [Ves85].

Conducting Critical Incident Interviews is quite labor intensive. As a result, the sample size is fairly small. With this sample we are able to perform an evaluation giving us a rich set of qualitative information. These initial results can be validated through further studies of larger samples using closed end survey instruments.

A biographical questionnaire is used to evaluate the subject pool. The questionnaire validates that subjects represent experienced rather than naive programmers, and that subjects include a valid cross-section of developers covering different language use, target applications, and development environments. The questionnaire requests information concerning education, on the job training, experience, languages used, and methods employed. We find that:

- 75% of the subjects are male; 25% are female. The 3 to 1 ratio is consistent with published reports that women constitute only 30% of the employed computer scientists [PPR+90].
- The mean age of the subjects is 33.45 years.
- The mean number of degrees held is 1.6. 65% of the subject hold a Bachelors degree as the highest degree, 30% hold a Masters degree, and one subject (5%) has a Ph.D.
- The mean number of training hours completed per subject in the two years preceding the study is 117.70 hours. The subjects reported a wide range of training hours.
- Subject responses to the question of "describe the software engineering methods and tools that you use now or in the past in your job" varied too greatly to be very useful.
- Subjects had worked in The Company a mean of 7 years in software engineering, ranging from 2 to 15 years.

The biographical data were analyzed for statistical significance at the .05 level when studied on a differential basis. That is, the data were split between *Exceptional* and *Non-Exceptional* subjects and compared. The *Fisher's Exact Test* is used to compare nominal variables with only two values (e.g. gender). The *t-test* is used to compare the means of ordinal values (*e.g. training hours*).

Since this was such a small sample, we did not expect any significant differences between the *Exceptional* and *Non-Exceptional* groups. However, *Years at Company in Software* are significantly related to *Exceptional Performance* with the 2-tail *t-test* calculated value of -3.21 with a significance level of .007. This significance demonstrates that although subjects were matched for total experience in the current organization, they were not matched for *Years at Company in Software*. Table 2 shows the differential information concerning years in The Company in software.

The demographic analysis indicates that, with the exception of the experience variable, no demographic data were significantly different between the exceptional and non-exceptional sub-samples in this small sample of 20 subjects. The lack of other statistically significant differences indicates experimental control of the other variables or speaks to the uniformity of the sample.

3 Critical Incident Interviews

The Critical Incident Technique attempts to discover the critical job requirements that have been demonstrated to make a difference between success and failure. The technique is based on two fundamental principles:

1. Reporting of facts regarding behavior is preferable to the collection of interpretations, ratings, and opinions based on general impression.

2. Reporting should be limited to those behaviors that, according to competent observers, make a significant contribution to the activity.

Flanagan provides an overview of the Critical Incident Technique for data collection [Fla54]. The technique was introduced during World War II in the Aviation Psychology Program to study combat leadership and pilot disorientation. The technique has since been refined and applied to measures of performance, measures of proficiency, training, selection, job design, equipment design, and leadership.

Protocol Analysis is used to translate the verbatim copy of an interview to a generalized set of cross-transcript results [ES84]. A formal process provides a record of the analysis and allows identified relations to be tied to specific utterances in the original transcripts [Web85, McC88]. The process is a movement from the specific to the general.

The process moves from transcripts to results in the following stages. Stage 1 converts an utterance to an observation by recognizing it as significant. A sentence or phrase is not included in the analysis until it is identified as being relevant to the research. The transcript is read carefully with the research question in mind to identify those utterances that must be identified and collected for later study.

Stage 2 develops the logical relationships that occur in the transcript. These relationships can be with the utterance itself, with the rest of the transcript or with previous literature. Stage 2 begins to attach meaning to and classify the utterance.

Stage 3 refines the observation in relation to all of the other Stage 2 observations in all of the transcripts. This stage moves from the study of one transcript to form relationships across transcripts.

In Stage 4, the researcher looks for patterns of inter-theme consistency and contradiction. Redundant themes are combined or eliminated. Themes that do not appear useful for the research question are eliminated.

Stage 5 identifies the patterns across the themes derived from the entire interview process.

3.1 Interview Process

Each Critical Incident Interview was conducted in a private room at the subject's work site. Each interview was tape-recorded, and the recordings were transcribed for later use. The interviews began with casual conversation followed by a description of the scope of the research and the general flow of the interview. The interview followed the basic structure and practices defined in [Hew89].

A typical interview began with an introduction similar to the following one taken from the transcript of one of the interviews:

> *What I'd like you to do is start off by thinking about a time which represents for you perhaps your personal best associated with software engineering in whatever form, so be it software development, software maintenance, testing, whatever it is, but a time at which you feel you were at your personal best, and when you've got one of those situations in mind, give me kind of a broad overview, a fifty word summary overview which is, how did you get involved in the situation, who were the other players, what was the nature of the task, and then we'll come back and we'll walk through it step by step in gory detail to find out exactly what you did in each case of that task.*

The subject would then describe an incident and the interviewer would probe for clarification or increased depth of response. The interviewer used probes, open-ended questions, questions of clarification, and reflective listening to keep the participant on the subjects of interest. The only way that the interviewer tried to direct the conversation was to provide additional clarification or to move on to other topics.

The subject generally described two to three significant incidents in the course of one interview. When each incident was completed, the subject was asked to describe the critical skill or competencies which were essential to the successful completion of the task. At the end of the discussion of the subject's incidents, the subject was asked to describe the list of essential competencies for an exceptional software engineer. The incidents formed one set of data regarding competencies, the self-description of skills formed a second set, and the manager generated competencies formed a third.

3.2 Interview Transcript Analysis

Data analysis of the Critical Incident Interviews used the Protocol Analysis technique of McCracken [McC88]. Each written transcript was reviewed and highlighted to identify tasks, incidents, competencies, self-described skills, and identified competencies for exceptional performance. Each transcript was reviewed individually to identify consistent themes which could be generalized as competencies for that individual. After each transcript was reviewed individually, the set of transcripts was examined to identify competencies which appear across multiple transcripts. These competencies were generalized and reworded as required to emphasize the similarities. Great care was taken not to over-generalize or distort the original meanings. A set of behaviors was identified based upon all of the the transcripts and served as a detailed explanation of the intent of the competency. At this point, original transcript text was retained and attached to the competency as further definition. A final pass allowed the combination of related competencies into a single competency.

All of the analysis to this point was done *blindly*. The transcripts were tagged with an identification number and the analyst did not know the name of the subject. Further, the analyst did not know if the transcripts were from an exceptional or non-exceptional subject.

The next step of the process was to count the number of subjects exhibiting an identified competency from each of the exceptional and non-exceptional groups. Those competencies exhibited by few subjects were dropped from further consideration. In general, at least three subjects had to identify a competency before it was retained. However, if one exceptional and one non-exceptional subject identified a competency, it was also retained.

The competencies identified from a subject's self-assessment of skills and from the subject's opinion of which competencies are related to exceptional performance were also identified and categorized. Those that appeared most often across transcripts were retained.

Finally, each manager who had provided subjects was asked to identify the competencies used in selecting the exceptional subjects for study. The manager was asked to list the skills, knowledge, or attributes that differentiated exceptional performers from non-exceptional performers in the study. The competencies identified most frequently across the five participating managers were retained.

4 Identified Competencies

Competencies are the skills, techniques, and attributes of job performance. Our analysis was directed towards identifying critical competencies of software engineers from three sources: subjects describing their own behavior, subjects reporting the competencies they think are related to exceptional performance, and managers describing the competencies of the subjects they selected as exceptional. The competencies from the three sources were merged into a single list of 38 competencies.

The 20 *Critical Incident Interviews* yielded a massive amount of data. Each interview lasted an average of two hours. Hence, the full set of data consists of 40 hours of taped interviews. The transcription of these tapes produced over 200,000 words for just the subject responses.

4.1 Derived Competencies

A total of 27 competencies were derived from the analysis of the subjects description of their own role in specific incidents. These competencies are identified by marking the skills, knowledge, or personal attributes alluded to while describing their own role in the incidents.

4.2 Self-Described Competencies

Subjects were asked to name the skills, knowledge, or personal attributes most important in helping them achieve their success in the described incident. The subjects were prompted for this response by a very open-ended question. Hence the replies are presumed to be the competencies considered most significant by the study participants.

Each subject enumerated those competencies that they felt most contributed to their own success. All summary lists for each of the 20 subjects were combined into a single list of competencies. Related competencies were merged to form a single competency. The number of subjects, both exceptional and non-exceptional, expressing the competency was noted. The competencies mentioned most frequently were retained for future analysis. Many of the competencies cited by engineers as being important to their own success, are, in fact, the same competencies identified from the analysis of the transcripts.

4.3 Manager Described Competencies

A third set of competencies was created by asking the managers of the subjects:

> *What Knowledge, Skills, or Attributes differentiate your exceptional performers from your non-exceptional performers?*

These are the same managers who classified the subjects in their organization as exceptional or non-exceptional. Sixteen differential competencies were identified by the five managers in the study. There was no further discussion with these managers to provide further elaboration on these competencies. Many of these competencies are similar to those identified by the analysis of transcripts or cited by engineers as those leading to exceptional performance.

4.4 Summary of Competencies

Table 3 summarizes the competencies identified most frequently from the multiple sources. The **Derived** category refers to those competencies extracted from the analysis of the interview transcripts. They represent those areas which the subject chose to discuss during their narration about their experiences. The number in this column records the number of subjects that described behaviors related to this competency. The **Self-Described** column records the number of subjects that offered the listed competencies when were asked to describe the skills, knowledge, and attributes associated with their successful performance on projects. The **Manager** records how many of the five managers cited the listed competencies as those that differentiate between exceptional and non-exceptional performers in their organization.

The competencies derived from the protocol analysis are considered to be more important than the competencies offered directly by the engineers or managers. This is because this study is based on the notion that behaviors associated with high performance are the unit of study. We consider competencies that are validated by multiple sources to be more important than competencies that come from only one source. A number of competencies were identified by the subjects and/or managers, but were not included in the set of competencies that will be used for further research. Some of these rejected competencies overlapped with those in Table 3. However, most were rejected because few people identified the competency, or it was not validated by multiple sources.

The identified competencies draw a broad picture of the necessary skills of a software engineer. The competencies can be organized into four categories, *Task Accomplishment, Personal Attributes, Situational Skills,* and *Interpersonal Skills.* The categories, and the behavior and/or attitudes of engineers that exhibit each competency are briefly described as follows:

1. Task Accomplishment Competencies:

   (a) Leverages/Reuses Code: pro-actively attempts to leverage other engineers' efforts by using their code or designs and attempts to leverage own effort by making newly developed code reusable.

Competency	Derived	Self-Described	Manager
1. Team Oriented	14	12	2
2. Seeks Help	11	4	
3. Helps Others	2	1	1
4. Use of Prototypes	14	3	
5. Writes/Automates Tests with Code	13		
6. Knowledge	13	12	
7. Obtains Necessary Training/Learning	12	7	
8. Leverages/Reuses Code	10		
9. Communication/ Uses Structured Techniques for Communication	8	8	
10. Methodical Problem Solving	9		
11. Use of New Methods or Tools	5		
12. Schedules and Estimates Well	4	2	1
13. Uses Code Reading	4		
14. Design Style	16		
15. Focus on User or Customer Needs	11	1	
16. Response to Schedule Pressure	9		
17. Emphasizes Elegant and Simple Solutions	8	2	
18. Pride in Quality and Productivity	12	1	
19. Pro-active/Initiator/Driver	11		
20. Pro-active Role with Management	10		
21. Driven by Desire to Contribute	8	5	
22. Sense of Fun	7		
23. Sense of Mission	6		
24. Lack of Ego	4		
25. Strength of Convictions	3	4	
26. Mixes Personal and Work Goals	3		
27. Willingness to Confront Others	3		
28. Thoroughness		4	
29. Skills/Techniques		11	
30. Thinking		9	
31. Desire to Do/Bias for Action		5	1
32. Attention to Detail		4	
33. Perseverance		13	
34. Innovation		4	
35. Experience		3	
36. Desire to Improve Things		3	
37. Quality		2	
38. Maintaining a "big picture" view/ Breadth of View & Influence		1	3

Table 3: Essential Competencies

(b) Methodical Problem Solving: uses a methodical approach (builds mental models, designs experiments, develops test tools, etc) in understanding and solving problems.

(c) Skills/Techniques: proficient in using design techniques, debugging skills; easily makes technology choices; good technical and software development background.

(d) Writes/Automates Tests with Code: applies incremental testing techniques during code development so that a given module achieves a high degree of reliability by the time it is completed.

(e) Experience: experience with a similar project.

(f) Obtains Necessary Training/Learning: actively seeks the necessary training required to complete the assigned task.

(g) Uses Code Reading: uses code reading and other group development techniques to ensure final code quality.

(h) Use of New Methods or Tools: seeks to improve performance or results through the use of new tools or methods.

(i) Schedules and Estimates Well: strong concern for schedules and estimates schedules well.

(j) Use of Prototypes: uses a prototyping method to assess key system parameters before designing the final product, and avoids using prototype as final implementation.

(k) Knowledge: at the time of assignment, possesses the unique skills or knowledge required to accomplish the task at hand.

(l) Communication/ Uses Structured Techniques for Communication: takes advantage of the tools and techniques of structured design in order to understand and communicate designs, but does not follow the complete formalism of the approach.

2. Personal Attributes Competencies:

(a) Driven by Desire to Contribute: values the sense of accomplishment which comes from making a direct contribution.

(b) Pride in Quality and Productivity: takes pride in producing defect free products on schedule in minimum time.

(c) Sense of Fun: enjoys the challenge of the assignment and the sense of accomplishment from completing it — has fun at work.

(d) Lack of Ego: stresses the solution over the source of the solution; does not care where a good idea comes from and does not feel the need to promote their own ideas.

(e) Perseverance: discipline, stubbornness, compulsiveness, dedication, and willingness to work hard on a task.

(f) Desire to Improve Things: not being satisfied with the status quo, setting high personal expectations and goals, and allowing time for improvement.

(g) Pro-active/Initiator/Driver: takes the initiative to complete important tasks. Influences others to consider alternative approaches.

(h) Maintaining "big picture" view/Breadth of View & Influence: sees the overall situation rather than focusing on details.

(i) Desire to Do/Bias for Action: sense of urgency, results oriented, and a willingness to try something.

(j) Thoroughness: makes sure all paths are covered — methodical, organized, and overcautious.

(k) Sense of Mission: driven by a sense of mission and clearly articulates goals to achieve a specific result.

(l) Strength of Convictions: exhibits and articulates strong beliefs and convictions. Acts in accordance with these beliefs, even when they are counter to specific managment direction.

(m) Mixes Personal and Work Goals: subjects find ways to align their project goals with their own personal development goals, and lobby their managers to receive the work assignments that match their personal desires.

(n) Pro-active Role with Management: pro-actively attempts to affect project direction by influencing management.

3. Situational Skills Competencies:

(a) Quality: a concern for reliability and a commitment to high quality.

(b) Focus on User or Customer Needs: considers customer or user input and feedback to be an essential ingredient in the design of products.

(c) Thinking: the ability to think algorithmically and structuredly.

(d) Emphasizes Elegant and Simple Solutions: creates solutions which are elegant and simple and allow for easy extension to furture needs.

(e) Innovation: having creative ideas.

(f) Attention to Detail: ability to deal with complexity.

(g) Design Style: use of design techniques relying on visual representation of designs; creates structured designs, usually without using formal techniques.

(h) Response to Schedule Pressure: in response to schedule pressure, sacrifices important parts of the design process.

4. Interpersonal Skills Competencies

(a) Seeks Help: pro-actively seeks the assistance of others in learning, researching, designing, understanding, debugging, or checking results.

(b) Helps Others: spends a significant amount of time assisting others in the completion of their tasks or influencing broad organizational direction.

(c) Team Oriented: values synergy of group efforts and invests the effort required to create group solutions even at the expense of individual results.

(d) Willingness to Confront Others: subjects will not let a conflict simmer and will openly confront another person in order to resolve a problem.

The competencies were analyzed on a differential basis using Fisher's Exact Test with a 2-tail probability. The score used for this test was the number of subjects that described behavior exhibiting a particular competencies. Only one of the competencies exhibited significant differences between exceptional and non-exceptional subjects. There was a significant differences between the groups with a 2-tail computed significance level of 0.0108 for the *Use of Prototypes* competency. We find that exceptional subjects are more likely to use prototypes to assess key system parameters. This result is especially noteworthy given the small sample size. None of the remaining competencies exhibited significance at the 0.05 level or better. Although most of the competencies cannot be used to distinguish between the exceptional and non-exceptional subjects, the derived competencies offer a unique view of the necessary skills of professional software engineers.

5 Related Work

Approaches for behavior-oriented software engineering research generally lie along a continuum between tightly controlled experiments (often with limited generality) and more broadly defined studies which stress qualitative psychological techniques [Shn80, Mor81, BSH86, Cur80, Cur87].

The bulk of the research to date favors the tightly controlled experimental approach. Studies seeking to correlate easily measured a priori factors with programmer performance have shown mixed results. In a study conducted by Evans and Simkins [ES89], 34 easily measured demographic, academic, experience, and behavioral variables could account for no more than 23% of the variation in student performance. On the other hand, Chrysler was able to explain over 85% of the variance in performance based on only thirteen program variables and five programmer variables [Chr78]. The subjects in Chrysler's study were experienced professional programmers rather than students. In another similar study, Moher and Schneider were able to explain 45-55% of the performance variability in student programmers, but for professional programmers only the years of experience was significant [MS81].

Our results on a small sample of professional programmers also found that the number of years of experience is the only statistically significant biographical factor. Rather than search for other simple predictors of performance, our major emphasis is on studying the actual behavior of software engineers when solving software engineering problems.

In behavioral experiments conducted at MCC, three experienced software developers were videotaped during the process of developing a design solution [GC87, GCK87]. The observed development process was not linear — designers operated simultaneously at various levels of abstraction and detail. Also, each designer exhibited a markedly different approach to design. Guindon describes the nonlinear design process as *serendipitous* or *opportunistic* [Gui88].

Of particular interest in the MCC studies is the use of an observational technique for gathering information. By observing the video tapes the researchers were able to obtain *thinking aloud reports*, and by collecting notes used in the designs were able to reconstruct the actual design sequence. The researchers also used *protocol analysis* to uncover cognitive factors at work in design. The major drawback to this study is its limited sample size.

Rather than directly observing behavior, our study analyses in-depth interviews of subjects describing their behavior. Although the incident interviews and transcript analysis used in our study require significant effort, they are far less labor intensive than the observational approach used in the MCC studies. As a result, we are able to examine a larger sample size than done at MCC.

6 Conclusions

We use the Critical Incident Interview technique in an in-depth review of 20 professional software engineers employed by a major computer firm. Our review includes an evaluation of biographical and Critical Incidence Interview data for 10 exceptional and 10 non-exceptional subjects. Our data shows that one biographical factor, *Years at Company in Software*, is significantly related to exceptional performance. We also analyze competencies identified by software managers. By combining the data obtained through the interviews and by the managers, we identify 38 essential competencies of software engineers. These competencies are shown in Table 3

The identified competencies provide an alternative view of the job of software engineering. Rather than an antiseptic application of formal software methods, we find a broad mix of knowledge, personality, and attitude involved. In addition to the expected skill competencies (*Use of Prototypes, Automates Tests, Reuses Code, Uses Code Reading, ...*) we find personality (*Sense of Fun, Lack of Ego, Willingness to Confront Others, Perseverance, ...*) and attitude (*Pride in Quality, Strength of Convictions, Bias for Action, Desire to Improve Things, ...*) emerge as significant factors in the engineering process.

The identification of competencies of software engineers is an important result, even if they are only *threshold* competencies. Threshold competencies are those competencies that are important to the job, and are exhibited equally by exceptional and non-exceptional performers.

The behavior of software engineers is of critical importance to the software engineering process. New methods to obtain such behavior data are important. Our results demonstrate the effectiveness of the Critical Incident Interview technique for collecting software engineering process data.

Our results can be strengthened by expanding the study to include engineers from more than one company. Corporate cultures can vary widely, and the definition of "good software engineering" differs between companies. Thus, the essential competencies are likely to be somewhat different in other companies.

Acknowledgement

We thank Charles Neidt, Kurt Olender, Jacek Walicki, and especially Gerry Johnson for their insight and guidance on this research. We thank James Turley for his significant contribution to the statistical analysis of the project data. We also thank the anonymous US corporation (referred to as *The Company*) for allowing us access to their software engineers for this study.

References

[Boe81] B. Boehm. *Software Engineering Economics.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[Boe83] B. Boehm. Seven basic principles of software engineering. *The Journal of Systems and Software*, 3(1):3–24, January 1983.

[Boe88] B. Boehm. Understanding and controlling software costs. *IEEE Trans. Software Engineering*, 14(10):1462–1477, October 1988.

[Bro87] F. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, April 1987.

[BSH86] V. Basils, R. Selby, and D. Hutchens. Experimentation in software engineering. *IEEE Trans. Software Engineering*, SE-12(7):733–743, July 1986.

[Chr78] E. Chrysler. Some basic determinants of computer programming productivity. *Communications of the ACM*, 21(6):472–483, June 1978.

[Cur80] B. Curtis. Measurement and experimentation in software engineering. *Proc. of the IEEE*, 68(9):1144–1157, September 1980.

[Cur81] B. Curtis. Substantiating programmer variablility. *Proc. of the IEEE*, 69(7):846, July 1981.

[Cur87] B. Curtis. Five paradigms in the psychology of programming. Technical Report STP-132-87, MCC, Austin, TX, April 1987.

[ES84] K.A. Ericsson and H.A. Simon. *Protocol Analysis: Verbal Reports as Data.* MIT Press, Cambridge, MA, 1984.

[ES89] G.E. Evans and M.G. Simkin. What best predicts computer proficiency? *Communications of the ACM*, 32(11):1322–1327, November 1989.

[Fla54] J. Flanagan. The critical incident technique. *Psychological Bulletin*, 51(4):327–358, July 1954.

[GC87] R. Guindon and B. Curtis. Control of cognitive processes during software design: What tools would support software designers? Technical Report STP-296-87, MCC, Austin, TX, August 1987.

[GCK87] R. Guindon, B. Curtis, and H. Krasner. A model of cognitive processes in software design: An analysis of breakdown in early design activities by individuals. Technical Report STP-283-87, MCC, Austin, TX, August 1987.

[Gui88] R. Guindon. A framework for building software development environments: System design as ill-structured problems and as an opportunistic process. Technical Report STP-298-88, MCC, Austin, TX, September 1988.

[Hew89] Hewlett-Packard Company, Corporate Training and Development. *The Horizon Project*, October 1989.

[McC88] G. McCracken. *The Long Interview,* Sage University Paper Series on Quantitative Applications in the Social Sciences, Vol 13. Sage Publications, Newbury Park, CA, 1988.

[Mor81] T.P. Moran. An applied psychology of the user. *ACM Computing Surveys*, 13(1):1–11, March 1981.

[MS81] T. Moher and G.M. Schneider. Methods for improving controlled experimentation in software engineering. *Proc. 5th Int. Conf. Software Engineering*, 1981.

[PPR+90] A. Pearl, M. Pollack, E. Riskin, B. Thomas, E. Wolf, and A. Wu. Becoming a computer scientist. *Communications of the ACM*, 33(11):47–57, November 1990.

[Shn80] B. Shneiderman. *Software Psychology: Human Factors in Computer and Information Systems.* Winthrop Publishers, Cambridge, MA, 1980.

[Ves85] I. Vessey. Expertise in debugging computer programs: A process analysis. *Int. J. Man-Machine Studies*, 23:459–494, 1985.

[Web85] R. Weber. *Basic Content Analysis,* Sage University Paper Series on Quantitative Applications in the Social Sciences, Vol 49. Sage Publications, Newbury Park, CA, 1985.