Department of Computer Science

Competencies of Exceptional and Non-Exceptional Software Engineers

Richard T. Turley and James M. Bieman

Technical Report CS-93-129

October 26, 1993

Colorado State University

Competencies of Exceptional and Non-Exceptional Software Engineers

Richard T. Turley
Colorado Memory Systems, Inc.

James M. Bieman
Colorado State University

Submitted for Publication

Colorado State Univ. Computer Science Technical Report CS–93–129
October 26, 1993

Key words: software engineering, large software development, software teams, knowledge and skills of software engineers, software productivity, software psychology

Abstract

The attributes of individual software engineers are perhaps the most important factors in determining the success of software development. Our goal is to identify the professional competencies that are most essential. In particular, we seek to identify the attributes that differentiate between exceptional and non-exceptional software engineers.

Phase 1 of our research is a qualitative study designed to identify competencies to be used in the quantitative analysis performed in Phase 2. In Phase 1, we conduct an in-depth review of ten exceptional and ten non-exceptional software engineers working for a major computing firm. We use biographical data and Myers-Briggs Type Indicator test results to characterize our sample. We conduct Critical Incident Interviews focusing on the subjects experience in software and identify 38 essential competencies of software engineers.

Phase 2 of this study surveys 129 software engineers to determine the competencies that are differential between exceptional and non-exceptional engineers. Years of experience in software is the only biographical predictor of performance. Analysis of the participants Q-Sort of the 38 competencies identified in Phase 1 reveals that nine of these competencies are differentially related to engineer performance using a t-test. A ten variable Canonical Discrimination Function consisting of three biographical variables and seven competencies is capable of correctly classifying 81% of the cases. The statistical analyses indicate that exceptional engineers (at the company studied) can be distinguished by behaviors associated with an external focus — behaviors directed at people or objects outside the individual. Exceptional engineers are more likely than non-exceptional engineers to maintain a "big picture", have a bias for action, be driven by a sense of mission, exhibit and articulate strong convictions, play a pro-active role with management, and help other engineers.

Authors addresses: R. Turley, Colorado Memory Systems, Inc., 800 S. Taft Ave., Loveland, CO 80537. Email: RICKTURL.COMEMSYS@CMS_SMTP.gr.hp.com, (303) 635-6490, Fax: (303) 635-6613; J. Bieman, Department of Computer Science, Colorado State University, Fort Collins, CO 80523. Email: bieman@cs.colostate.edu, (303)491-7096, Fax: (303) 491-6639.

Direct correspondence concerning this paper to: J. Bieman, Department of Computer Science, Colorado State University, Fort Collins, CO 80523, bieman@cs.colostate.edu, (303)491-7096, Fax: (303)491-6639.

1 Introduction

We report on a study of the differences between individual software developers. 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. Further, once these skills are recognized, we hope that they can be transferred to the software engineering community at large through formal training programs [1]. Thus, additional software engineers can be taught these valuable skills. Our overall goal is to identify the skills, techniques, and attributes that differentiate between exceptional and non-exceptional software engineering performance.

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 [2]. By studying exceptional programmers, the individual capabilities that most influence performance can be identified [3].

Most research into the development of software focuses on the individual only to the extent that individuals are members of a larger development effort. Although the team is a critical component in software development, most research misses a fundamental opportunity to identify and exploit the proven ability of highly talented individual contributors. Weinberg noted the lack of research on individuals observing that "Our profession suffers under an enormous burden of myths and half-truths." [4]. The industry has a great lore about the factors affecting software productivity, but few facts are known.

Bohem 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 [5]. If the personal attributes of these most productive individuals can be understood, a number of exciting opportunities present themselves:

- Understanding the characteristics of the most successful software developers could lead to the improvement of *all* software developers.
- Once the characteristics are understood, it may be possible to develop specific toolsets and aids to further increase the productivity of these individuals.
- A valuable criterion of the selection of software developers may be discovered.

Brooks suggests the "use of great designers" as one of five promising approaches to improve software development productivity [6]. One of Boehm's seven basic principles of software engineering is to use "better and fewer people" [7].

Typical experimental approaches to studying individuals in software development start with an individual's experience and prejudices about software development [8]. A technique for improvement is proposed, implemented and tested [9, 10]. The results of these experiments are then analyzed and often valuable results are achieved.

This study follows a different approach. We start with professional software developers who are acknowledged for their software ability. Our focus on the top individual contributors breaks with the traditional emphasis on the team. We seek to enhance the value of teams by ensuring that each individual is operating at peak productivity.

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 a two phase study designed to determine the essential competencies of professional software engineers [11]. In Phase 1 we identify these competencies via the Critical Incident Interview technique. In Phase 2

Phase 1 Population Summary	Total
#Engineers	252
#SW Engineers	150
#Study Participants	20
# of Exceptional SW Engineers Studied	10
% Studied Exceptional SW Engineers (out of total SW Engineers)	6.7%

Table 1: Phase 1 Population Summary

we differentially relate these competencies to engineer performance. Phase 1 corresponds to the *qualitative* portion of the research in which the competencies associated with the job of software engineering are first uncovered. Phase 2 corresponds to the *quantitative* portion of the research in which the competencies discovered in Phase 1 are validated and considered on a differential basis between exceptional and non-exceptional performers.

The remainder of this paper is organized as follows. Section 2 describes Phase 1 in detail, and Section 3 presents the details of Phase 2. Section 4 contains a discussion of the implications of the results. We review related work in Section 5, and our conclusions are given in Section 6.

2 Phase 1

In Phase 1, we identify critical professional competencies through an in-depth analysis of a small sample of exceptional and non-exceptional software engineers. We use a biographical questionnaire and a Meyers-Briggs Type Indicator (MBTI) test [12] to characterize our sample. We conduct Critical Incident Interviews to identify the significant competencies of software engineering.

2.1 Phase 1 Subjects

Subjects are drawn from five commercial research and development laboratories at three different sites of a single company. 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 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 Phase 1 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 *% Studied Exceptional SW Engineers* 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 non-exceptional 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 [13].

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.

2.2 Biographical Profile

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 [14].
- 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%) earned a Ph.D.
- The mean number of training hours completed per subject in the two years preceding the study is 117.70 hours. Completed training ranged from zero to 306 contact 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 at The Company a mean of 7 years in software engineering, ranging from 2 to 15 years.

The data were split between *Exceptional* and *Non-Exceptional* subjects and compared. The biographical data were analyzed for statistical significance at the .05 level when studied on a differential basis. The *Fisher's Exact Test* was used to compare nominal variables with only two values (e.g. gender). The *t-test* was 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 subsamples in this small sample of 20 subjects. The lack of other statistically significant differences indicates experimental control of the other variables, the uniformity of the sample, or the weakness of the identified differences.

(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

2.3 Myers-Briggs Cognitive Style Type Indicator (MBTI)

The MBTI is a tool for determining psychological type [12, 15]. We use it to determine if type differences exist between exceptional and non-exceptional engineers.

Through a questionnaire, the MBTI computes a score for four contrasting personality pairs:

- extrovert vs. introvert
- sensing vs. intuitive
- thinking vs. feeling
- judging vs. perceptive

The purpose of the MBTI is to identify, from self-report of easily recognized reactions, the basic preferences of people with regard to perception and judgement [16]. The four preferences are assumed to interact in complex nonlinear ways to produce one of 16 psychological *types* with different attributes [17]. The MBTI can provide a continuous score for each of the four preference scales allowing for statistical analysis of significant differences [12]. A detailed description of the MBTI appears in [15].

All 20 subjects completed the *Myers-Briggs Type Indicator (MBTI)* test. Figure 1 shows the distribution of the Phase 1 study participants according to one of 16 personality types. Eighteen out of twenty subjects exhibit the *Introvert* type. The *Introvert* tendency is consistent with the Kagan-Douthat study of students learning Fortran which found a tendency towards introversion in higher performing programming students [18]. We also found a that 17 out of 20 subjects exhibited the *Thinking* type (rather than *Feeling* type). This result is consistent with broader studies that find that computer specialists exhibit the thinking preference 67% of the time [12].

The MBTI test uncovered an interesting tendency for exceptional engineers to favor the *Introvert, Thinking* type. The most frequent classification for exceptional performers is the **INTJ** (*Introvert, Intuitive, Thinking, Judging*) type. The **INTJ** type occurs in only 10% of the male college graduates [12]. Hence these exceptional engineers differ from the (male) population at large.[1] Only one of the exceptional engineers is classified as an *Extrovert* type.

The non-exceptional engineers exhibited more varied personality types. Only two of the non-exceptional engineers exhibited the **INTJ** type. Six of the non-exceptional engineers exhibited a combination of the *Introvert* and *Judging* types. Like the exceptional group, only one of the non-exceptional engineers is classified as an *Extrovert* type.

A statistical analysis via the *t-test* of the differential MBTI scores revealed that there were no significant differences between the scores of the exceptional and non-exceptional subgroups. This indicates that either personally type is not a good predictor of performance or that the sample size is too small.

[1] The data given by Myers and McCaulley [12] is unfortunately limited to males.

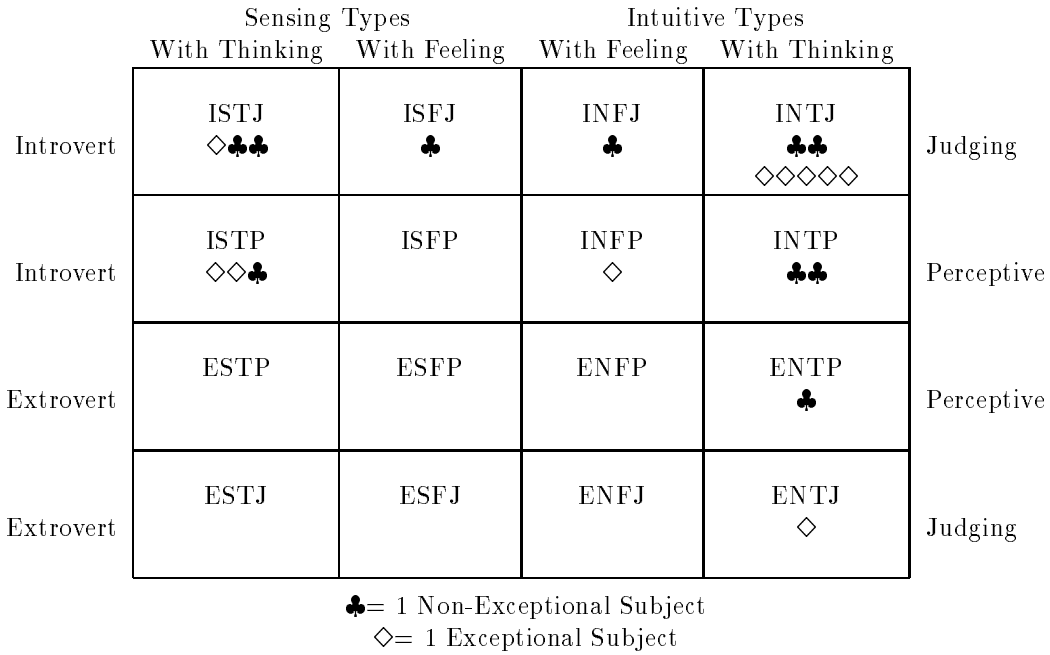

Figure 1: Myers-Briggs Type Indicator (MBTI) Results.

We do not use the MBTI test in Phase 2 because of the inconclusive results in Phase 1 (due to the small sample), and because the MBTI test is expensive and very time consuming for a larger sample.

2.4 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 [19].

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 two hour 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.

2.5 Analysis of 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 [20]. 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 [21]. A formal process provides a record of the analysis and allows identified relations to be tied to specific utterances in the original transcripts [22, 23].

We used the Protocol Analysis technique described by McCracken [23]. 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.

2.6 Identified 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.

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.

Self-Described Competencies

Subjects were also 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.

Manager Described Competencies

Another set of competencies was created by asking the managers of the subjects:

> *What are the Knowledge, Skills, or Attributes that 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.

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. And it is through the interviews that subjects demonstrate these behaviors. 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. These competencies were rejected because few people identified the competency, or it was not validated by multiple sources.

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 technical skill competencies (*Use of Prototypes, Automates Tests, Reuses Code, Uses Code Reading, ...*) we find personality (*Sense*

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
	$n = 20$	$n = 20$	$n = 5$

Numbers indicate the frequency that competencies are identified as follows:
Derived: extracted from interview transcripts.
Self-Described: offered by subjects as most important competencies.
Manager: offered by managers as differential competencies.

Table 3: Essential Competencies

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 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 competency. 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 based on this small sample of 20 subjects, the derived competencies offer a unique view of the necessary skills of professional software engineers. For complete a description of all of the identified competencies see [11].

3 Phase 2

In Phase 2, we use the identified competencies, larger samples, and objective survey instruments to detect significant differences between exceptional and non-exceptional software engineers. Our objectives are to determine which competencies identified in Phase 1 are differentially related to performance, and determine if a simple predictor of performance exists. We develop a predictive model that uses the competencies to predict whether a particular engineer will be ranked as exceptional or non-exceptional.

3.1 Phase 2 Subjects

In Phase 2, we seek to validate the Phase 1 results against a broader population. Thus we expand the sample of exceptional and non-exceptional software engineers both in quantity and diversity. Matching for time in the organization is not required since the breadth of subjects is expected to eliminate the relevance of differences in experience. In addition, the definition of "exceptional" was widened to include the top 30% rather than the top 5% of engineers. This widened definition allows a more even mix of exceptional and non-exceptional engineers in the study. Allowing more subjects to be defined as exceptional is a conservative approach — we increase the risk that a competency will not be identified as differential. The resulting increase in the relative number of exceptional engineers in the subject pools also aids the statistical analysis.

As in Phase 1, all subjects are software engineers employed by The Company. We did not use a two year minimum experience criterion as in Phase 1. Managers were asked to distribute surveys to their entire lab on a differential basis — 70% of the surveys are distributed to non-exceptional performers and 30% to exceptional performers. The determination of exceptional versus non-exceptional was again made by the managers. Managers were allowed to distribute exceptional surveys to slightly more than 30% of their lab based on their judgment of performance. Managers were instructed to keep the differential nature of the survey confidential. A total of 275 survey instruments were distributed to engineers working in nine divisions of The Company at three sites. The engineers participate in the development of five types of software applications — test & measurement, embedded firmware, CAE/CAD/CASE software, graphics, and operating systems.

Population Summary	Total
Surveys Distributed	275
Total Responses	129
Response Rate	46.9%
# Exceptional Responses	41
# Non-Exceptional Responses	88
% Exceptional	31.8%
% Non-Exceptional	68.2%

Table 4: Phase 2 Population Summary

3.2 Descriptive Statistics

Each survey packet contained a letter of instruction that outlined the assignment and clearly indicated the voluntary nature of the study. Each packet included a Biographical Questionnaire and a set of Q-Sort cards. The packet also included a pre-addressed return envelope for returning the completed survey. The results were thus *blind* in that we did not know the names of study participants or their corresponding rating.

The Biographical Questionnaire used in Phase 2 was nearly identical to the one used in Phase 1. Some minor changes were made for book keeping purposes. A *Results of Sorting* section was added to capture the results of the Q-Sort activity.

The total number of surveys distributed, responses, and distribution between responses from exceptional and non-exceptional engineers are indicated in Table 4. Only four of the Phase 2 responses were incomplete in some of the major independent variables. We consider only valid surveys in our analysis of each variable. As a result, there is some variation in the reported number of samples n. The response rate of nearly 50% indicates the level of interest in this information at the company studied. The sample of 129 participants provides sufficient statistical power to complete the study. The response rate for exceptional and non-exceptional performers was similar, since 30% of the surveys were distributed to exceptional performers and 31.8% of those returned were from this group.

We collected descriptive statistics using a Biographical Questionnaire to ensure that the Phase 2 sample is similar to that of Phase 1 We also analyzed the data for normalcy so that subsequent statistical steps will be valid. Phase 2 subjects can be described as follows:

- 78.9% of the subjects are male; 21.1% are female. This distribution is similar to the Phase 1 mix and reflects the preponderance of males in Computer Science.
- The mean age is 32.45 years and is comparable to the Phase 1 mean age of 33.45 years.
- The subject pool is well educated with over 53% holding two or more degrees, and 40% have a Master's degree as their highest degree. Over 74% hold at least one degree in Computer Science, while only 35% of Phase 1 subjects held degrees in Computer Science.
- The mean number of training hours completed in the prior two years was 102, versus 117 hours for Phase 1 subjects.
- Subjects had worked in The Company for a mean of 6 years in software engineering, compared to a mean of 7 years for Phase 1 participants.

We analyzed the descriptive data to determine if there are significant differences between the exceptional and non-exceptional groups. The Chi-Square test is used to test for statistically significant differences between nominal variables such as gender. The t-test is used to test for significant differences between ordinal variables. Differences are considered statistically significant when the calculated 2-tail significance level is less than 0.05. We found only three statistically significant biographical variables associated with exceptional performance: *Years at Company in Software, Total Years in Software,* and *Total Years Worked.* This is consistent with the Phase 1 result indicating that *Years at Company in Software* is a differential attribute.

3.3 Q-Sort

The Q-Sort method is used to assist subjects in ranking the competencies identified in Phase 1. Q Methodology encompasses the Q-Sorting Technique, which is designed to provide practical means for subjects to sort and researchers to analyze large lists of items [24]. The method stresses the individual's perception of *value* in a set of statements as the actual data under study. The technique has a long history being first promoted by Stephenson in the 1930's. His text continues to be a significant reference on the technique [25].

Using Q-Sort, a subject is asked to rank order a set of items against a specific condition of instruction. The ordering is *quasi-normal* in that it asks subjects to place the item in one of a limited number of bins or piles. The number of items is expected to far exceed the number of piles. Each pile maintains a specific relationship to the other piles. The number of items to be placed in each pile is meant to be proportional to a roughly normal distribution of the items. For example, if there are ten items to distribute across five piles, the first pile will have one item, the second pile will have two items, the third pile will have four items, the fourth pile will have two items, and the fifth pile will have one item. This arrangement approximates a normal distribution.

Critical to the sorting is the *condition of instruction.* A subject may provide a radically different sorting based upon the instructions given. For example, a subject could be instructed to sort competencies based upon (1) the order which most relates to being exceptional, or (2) the order based on the subjects own behavior on the job. We would expect a different result depending on which instructions are given.

We emphasize that the criterion for sorting the competency cards is the subject's self report of his or her own behavior.

Using Q-Methodology, a Q-Sort task is normally completed by a subject with the help of the researcher. We used a simplified approach to the Q-Sorting task to allow subjects to complete the task on their own. Each subject received a set of *Competency Cards* with one competency listed on each of 38 $3'' \times 5''$ index cards. (Figure 2 shows a competency card for Competency #1, *Team Oriented.*) A set of *Pile Marker Cards* is also included in order to prompt subjects to create the correct number of piles and to include the correct number of cards in each pile. Further, the Pile Marker Cards include prompts to remind subjects of the definition of the continuum across which the competencies are sorted. The directions that subjects followed in completing the Q-sorting exercise are given in Figure 3.

Study participants sorted a set of 38 competencies into a quasi-normal distribution of seven piles. Each pile was assigned an integer value from zero to six. Zero means *Least Like My Behavior* while six means *Most Like My Behavior.* For each survey, the Q-Sort item was assigned the integer value associated with the pile that the subject placed it into. We calculated the mean Q-Sort for the full sample of both exceptional and non-exceptional engineers. We also calculated the skew and kurtosis numbers which indicate that all Q-Sort items are normally distributed.

A t-test comparison of means for each of the Q-Sort Competencies is given in Table 5.

(Items sorted by mean score of exceptional responses)

Delta	Competency	XP Mean (n=40)	NXP Mean (n=85)	Test Value	Sig. Level (2 Tail)
0	Concern for Reliability/Quality	4.050	4.224	0.68	0.497
0	Focus on User/Customer Needs	4.025	3.859	-0.66	0.512
0	Thinking	3.925	3.859	-0.25	0.804
+2	Pride in Quality/Productivity	3.750	3.576	-0.70	0.488
+2	Emphasizes Elegant & Simple Solutions	3.725	3.518	-0.85	0.395
-2	**Driven by Desire to Contribute**	**3.550**	**4.012**	**2.20**	**0.029****
+8	**Mastery of Skills/Techniques**	**3.500**	**2.965**	**-2.22**	**0.028****
+20	**Helps Others**	**3.500**	**2.188**	**-4.99**	**0.000****
-1	Innovative	3.425	3.435	0.04	0.966
+12	**Maintains "big picture" view**	**3.425**	**2.647**	**-2.66**	**0.009****
-2	Enjoys Challenge of Assignment – Has Fun	3.325	3.435	0.38	0.707
-7	**Seeks Help From Others**	**3.325**	**3.835**	**2.10**	**0.038****
0	Lack of Ego	3.250	3.259	0.04	0.966
+6	Prior Experience	3.250	2.824	-1.67	0.097
-5	Attention to Detail	3.225	3.412	0.65	0.519
+3	Pro-active/Initiator/Driver	3.100	2.929	-0.82	0.414
-3	Team Oriented	3.050	3.235	0.65	0.515
-7	Leverages/Reuses Code	3.050	3.435	1.56	0.120
-2	Desire to Improve Things	3.050	3.141	0.40	0.687
-8	Perseverance	3.000	3.447	1.63	0.107
+9	**Strength of Convictions**	**2.975**	**2.341**	**-2.04**	**0.044****
+13	**Pro-active Role with Management**	**2.925**	**2.035**	**-2.95**	**0.004****
+4	Schedules and Estimates Well	2.900	2.471	-1.56	0.122
-8	Methodical Problem Solving	2.900	3.235	1.58	0.117
-7	Writes/Automates Tests with Code	2.775	3.224	1.66	0.099
+6	Driven by a Sense of Mission	2.750	3.388	-1.61	0.111
-3	Use of New Methods or Tools	2.700	2.871	0.66	0.513
-3	Uses Decomposition Design Style	2.700	2.765	0.25	0.805
-3	Desire to Do/Bias for Action	2.675	2.588	-0.28	0.777
-9	Obtains Necessary Training/Learning	2.600	3.035	1.86	0.065
-8	Uses Code Reading	2.550	2.965	1.59	0.115
-1	Use of Prototypes	2.550	2.459	-0.33	0.740
+3	Possesses Unique Knowledge	2.325	2.094	-0.97	0.332
0	Mixes Personal and Work Goals	2.250	2.529	1.11	0.270
-6	Thoroughness - Methodical, Cautious	2.150	2.753	1.89	0.061
-5	**Willingness to Confront Others**	**2.125**	**2.706**	**2.23**	**0.027****
+1	Structured Techniques for Communication	2.050	2.012	-0.14	0.890
-1	**Responds to Schedule Pressure by Sacrificing Parts of Design Process**	**1.600**	**2.294**	**2.29**	**0.024****

XP = Exceptional Subject, NXP = Non-Exceptional Subject.
Differences are considered statistically significant when the calculated significance is less than 0.05.
These instances are in bold print and are denoted by "**".

Table 5: Differential Q-Sort Competency Responses, T-Test Results

DEFINITION
I value the synergy of group efforts and invest the effort required to create group solutions, even at the expense of my individual results.

Key Behaviors

- I balance the strengths and weaknesses of other team members.
- I promote constant communication among team members using techniques such as brainstorming sessions, travel, phone calls, e-mail, or just being physically close to the rest of the team.
- I recognize synergy of group efforts and invest personal time and energy to leverage it.

Item #1

Figure 2: Q-Sort Card for Competency #1, Team Oriented.

The means are calculated separately for exceptional and non-exceptional performance and tested for difference. The two means are considered different when the calculated significance level is less than 0.05. These entries are denoted by ** in Table 5. The table is sorted by the mean scores of the exceptional responses. The *Delta* column represents the number of places that a particular competency moves in its rank order when sorted by exceptional means rather than sorted by the full sample means.

Nine competencies show statistically significant differences in the mean values reported by the exceptional and non-exceptional engineers. Thus 24% of the 38 competencies are related to the difference in performance of exceptional and non-exceptional engineers. The five competencies which have a higher mean for exceptional performers and the behavior and/or attitudes of engineers that exhibit each competency are briefly described as follows:

1. Helps Others: spends a significant amount of time assisting others in the completion of their tasks or influencing broad organizational direction. These engineers act as lab-wide consultants for process or product issues; they review, direct, or influence the work of other engineers; they teach engineering skills to other engineers.

2. Pro-active Role with Management: pro-actively attempt to affect project direction by influencing management. These engineers discuss issues concerning other engineers with their managers; they attempt to set project direction and make project decisions by influencing their managers; they promote product ideas through demos or selling of ideas to management.

3. Exhibits and Articulates Strong Convictions: exhibits and articulates strong beliefs and convictions, and acts in accordance with these beliefs, even when they are counter to specific management direction. These engineers act in accordance with their beliefs rather than acting solely on their assignment; they risk their performance ranking in an effort to secure the best solution; they argue forcefully for a specific point of view.

4. Mastery of Skills and Techniques: mastered the skills and techniques necessary for good software design and implementation. These engineers have a strong technical and software

Competency Sorting Exercise

The objective of this exercise is to determine which job competencies identified in Phase 1 research best characterize the Company's Software Engineering population. You will sort these competencies based on how well they describe your behaviors on the job, especially when you're performing at your best. Try to think of the best software experience you've had and use that to guide selection of which attributes best describe your behavior on the job.

1. Be sure that you have a clear desk or table to work on before you start. You will be placing 3×5 cards in one of 7 piles so you need space to spread these out. Find the supplied pile markers in the envelope and lay these out on your table in order from number 6 on your left to number 0 on your right. These pile markers are annotated to remind you that column 6 represents those competencies that are most like your behavior and column 0 represents those competencies that are least like your behavior.
2. Read through all 38 competency cards to become familiar with them.
3. Sort all of the cards into 3 piles of any number of cards. Place to the left the cards which include the competencies which best describe your behavior in the process of software engineering. Place to the right those cards which include competencies which least describe your behavior in the process of software engineering. Place those cards with competencies about which you are unsure in the middle pile.
4. During the sorting you will spread the items in piles under the pile markers, while maintaining the general left-center-right relationships.
5. Select the 2 items that most strongly relate to your behavior on the job as a software engineer. Think in particular about those time which have been a personal best for you. Place these two cards under the column marker labeled 6. The order of these cards under the marker is not important. All will receive the same score.
6. Now select the 2 items that least reflect your behavior on the job as a software engineer. Place these under the column marker labeled 0.
7. Continue in this way, alternating between the left and right sides of the distribution, placing the indicated number of cards below each column marker. Feel free to move any card at any time should you change your mind about which competencies are most closely related to your actual behavior. All that matters is that the right number of cards eventually are found beneath each column marker. Try not to take too long agonizing over the placement of any one card. Your first impulse for placing the card is probably the best. If it helps, you can jot a short phrase that captures the essence of the competency directly onto the card as a prompt to use in sorting.
8. Review your groupings to be sure that they accurately reflect your behavior while completing your software engineering assignments. Move any cards you wish to better reflect which competencies most apply to you doing your job. Now record the item identification numbers found in the lower right hand corner of each card in the appropriate column on the back of the **Biographical Questionnaire**.

If you have any questions, don't hesitate to give me a call at 123-4567 to ask for help.

Figure 3: Q-Sorting Instructions given to Company Participants.

development background; They are comfortable with multiple software design and implementation techniques; they have very strong software development skills.

5. Maintains "Big Picture" View: sees the overall situation rather than focusing on details in an attempt to influence the project direction. These engineers remain aware of what other engineers are doing and suggest ways to better achieve project objectives; they try to be sure that project goals make sense, and work to change them if necessary; they try to fit their project into the broader scheme of division programs.

The four competencies which have a higher mean for non-exceptional performers are:

1. Seeks Help from Others: pro-actively seeks the assistance of others in learning, researching, designing, understanding, debugging, or checking results. These engineers ask previous developers to explain their designs; they ask other engineers to critique or evaluate their designs; they survey others to create lists of alternatives.

2. Responds to Schedule Pressure by Sacrificing Parts of Design Process: In response to schedule pressure, these engineers are forced to provide incomplete documentation; they do not have time to adequately inspect or test the product; they will not prototype or adequately design risky parts of the product.

3. Driven by Desire to Contribute: values the sense of accomplishment which comes from making a direct contribution. These engineers seek assingments where they can contribute and feel rewarded by the chance to contribute.

4. Willingness to Confront Others: confront others when necessary to ensure a good design or product solution. These engineers will not let a conflict simmer and will openly confront another person in order to resolve a problem; they will raise a tough issue of conflict with another engineer to their manager in an effort to have the conflict resolved.

Of particular interest, the *Use of Prototypes* is not differential according to the Q-Sort, although it was differential in Phase 1 with a significance level of 0.0108. We offer two possible explanations for this discrepancy. A 0.0108 significance level means that there is a 1% chance that the relationship found in Phase 1 was just the result of chance, and *Use of Prototypes* was not really differential. A more likely explanation is that the criteria for determining exceptional performance were significantly different between Phase 1 and Phase 2. In Phase 1 the exceptional group was to be in the top 5% of the organization, while in Phase 2 the exceptional group included the top 30%. The differences between the exceptional and non-exceptional groups are likely to be diminished by relaxing of standards for selecting exceptional engineers.

Both exceptional and non-exceptional engineers indicate that they do not respond to schedule pressure by sacrificing parts of the design process. The *Responds to Schedule Pressure* competency was ranked 38th — last — by exceptional engineers and 37th — next to last — by non-exceptional engineers. Although both groups ranked this competency very low, the difference in ranking proved to be statistically significant. Non-exceptional engineers are more likely to provide inadequate documentation or inadequate testing when the schedule gets tight. Somehow the exceptional engineers are able to avoid this trap.

A few of the results are counter-intuitive. For example, there is not a significant difference between the exceptional and non-exceptional groups in their use of new methods and tools and their view of the role of innovation. In both cases the non-exceptional engineers ranked these competencies slightly higher than the exceptional group. A possible explanation for the counter-intuitive results is that there may be a discrepancy between how engineers view their own activities

VARIABLE
Gender
Highest Degree Held
Computer Science Degree?
Engineering Degree?
Math Degree?
Training Hours
Total Years in Software
Total Number of Languages
38 Competencies

Table 6: Retained Variables for Discriminant Analysis

and an evaluation of an outside observer. Exceptional and non-exceptional engineers may view themselves equally in terms of innovation as described in the competency cards:

> "I am innovative in my solutions to problems. I like to create alternatives that are both creative and practical. I have creative ideas and solutions to problems."

Yet an outside observer might rank the exceptional and non-exceptional engineers quite differently. Both the exceptional and non-exceptional engineers engineers rank innovation highly — innovation is ranked 9th by the exceptional engineers and 8th by the non-exceptional engineers (out of 38 competencies).

3.4 Discriminant Analysis

We performed a discriminant analysis of the full set of non-correlating variables, and then perform the analysis using a set of fewer variables. First we analyzed the correlations of variables, since highly correlated variables cannot be used in the analysis.

Cross-correlations of the biographical variables demonstrates that age and experience variables are highly correlated. This is expected since engineers who are older will tend to have more experience. We assume that experience rather than age is the important variable here. Since not all of these variables can be used in subsequent analysis, we select *Total Years in Software* as the most appropriate variable. This choice is consisting with prior literature [26]. There is also a natural high correlation between the *Number of Degrees Competed* and the *Highest Degree Completed*. We use the *Highest Degree Completed* in subsequent analysis. None of the 38 competency variables were correlated with each other or with the biographical variables at a level of 0.60 or better. Hence all are used in the subsequent discriminant analysis. The variables that will be used in the discriminant analysis are shown in Table 6. These variables were entered into a stepwise discriminant analysis using a 24 step process with the results shown in Table 7.

Table 7 shows that 49% of the variance $(1 - \lambda)$ can be explained by the 20 variables in the Canonical Discriminant Function following the analysis. A more significant result is demonstrated in Table 8 where we find that the function composed of the 20 variables in Table 7 is able to correctly classify 86% of the cases collected in this study.

As a practical refinement, the discriminant analysis was rerun over the same variables, but only allowing the first 10 variables of Table 7 to enter the Canonical Discriminant Function. This was an attempt to create a more tractable predictor function which can be more readily used in

Step	Action Add	Action Delete	Vars In	Wilks' Lambda	Sig.	Competency (C) or Biographical Variable (B)
1	√		1	.83854	.0000	Helps others (C)
2	√		2	.76331	.0000	Total years Software experience (B)
3	√		3	.72147	.0000	Driven by bias for action/urgency (C)
4	√		4	.69362	.0000	Total languages used professionally (B)
5	√		5	.67090	.0000	Willingness to confront others (C)
6	√		6	.64644	.0000	Exhibits and articulates strong convictions (C)
7	√		7	.62845	.0000	Perseverance (C)
8	√		8	.61493	.0000	Driven by sense of mission (C)
9	√		9	.59993	.0000	Responds to schedule pressure (C)
10	√		10	.58924	.0000	Math Degree Held? (B)
11	√		11	.57852	.0000	Uses prototypes to assess design (C)
12	√		12	.56804	.0000	Schedules and estimates well (C)
13	√		13	.55790	.0000	Maintains "big picture view" (C)
14	√		14	.55016	.0000	Uses structured techniques for communication (C)
15	√		15	.54255	.0000	Team oriented (C)
16	√		16	.53526	.0000	Engineering degree held? (B)
17	√		17	.52980	.0000	Takes pride in quality and productivity (C)
18	√		18	.52393	.0000	Total training hours (C)
19	√		19	.51692	.0000	Uses code reading (C)
20	√		20	.51101	.0000	Focuses on user or customer needs (C)
21		√	19	.51579	.0000	Maintains "big picture view" (C)
22		√	18	.51986	.0000	Uses prototypes to assess design (C)
23	√		19	.51465	.0000	Writes/automates tests in parallel (C)
24	√		20	.50901	.0000	Uses methodical problem solving approaches (C)

Table 7: Full Discriminant Analysis — Summary Table

Actual Group	No. of Cases	Predicted Group Membership 0	1
Group 0	83	73	10
Non-exceptional		88.0%	12.0%
Group 1	40	7	33
Exceptional		17.5%	82.5%

% of "grouped" cases correctly classified:	86.18%	
125	Cases were processed.	
0	Cases were excluded for missing or out-of-range group codes.	
2	Cases had at least one missing discriminating variable	
123	Cases were used for printed output	

Table 8: Full Discriminant Analysis — Classification Results

Actual Group	No. of Cases	Predicted Group Membership 0	1
Group 0	83	68	15
Non-exceptional		81.9%	18.1%
Group 1	40	8	32
Exceptional		20.0%	80.0%
% of "grouped" cases correctly classified:		81.3%	
	125	Cases were processed.	
	0	Cases were excluded for missing or out-of-range group codes.	
	2	Cases had at least one missing discriminating variable	
	123	Cases were used for printed output	

Table 9: Limited 10 Variable Discriminant Analysis — Classification Results

practice. The full discriminant analysis in Table 7 shows that after the first 13 variables entered the discriminant function, subsequent variables explained less than 1% of the remaining variance. Thus we find a practical cutoff for additional variables. Further, the eleventh and thirteenth variables entered (*Uses Prototypes* and *Maintains "big picture" view* competencies) were subsequently removed from analysis. This is an indication that the eleventh, twelfth, and thirteenth variables are not important to retain for further analysis. Hence the analysis included only the first 10 variables in the Canonical Discriminant Function. Table 9 gives the classification results for the reduced case of 10 variables. The function of 10 variables is able to correctly classify over 81% of the cases collected in this study.

The ten variable function is nearly as effective as the full twenty variable function in classifying the exceptional and non-exceptional cases. The total variance explained by these ten variables is 41%. The *Helps Others* competency explains 16% of the variance in the sample. The *Total Years Software Experience* variable explains another 8% of the variance. The *Bias for Action* competency explains 4% of the sample variance. Each of the seven remaining variables explains less than 3% of the variance of the sample.

In the ten variable function, seven of the variables are competencies; the remaining three are biographical variables. Four of the competencies in the function were found as differential using the t-test. Thus, three of the competencies are differential in the ten variable Canonical Discriminant Function, but are not differential using the t-test. These three differential competencies and the behavior and/or attitudes of engineers that exhibit each competency are briefly described as follows:

Two of these differential competencies have a higher mean for exceptional performers:

1. Desire to Do/Bias for Action: driven by a bias for action and sense of urgency in completing assignments. When faced with a tough problem, these engineers do not hesitate to get started and develop the required capabilities as they go; they are results oriented and want to make progress on a regular basis; they push themselves to achieve results quickly.
2. Sense of Mission: driven by a sense of mission and clearly articulate goals to achieve a specific result. These engineers create and articulate clear and specific goal statements; they drive the project to achieve specific goals.

One of these differential competencies has a higher mean for non-exceptional performers:

1. Perseverance: methodical, organized, and cautious in their work. These engineers make sure that all paths are covered in their design and problem solving; they work slowly and carefully to avoid making mistakes.

4 Discussion

We conclude from our evaluation of Phase 2 results that experience is indeed a significant predictor of performance. This is particularly true when the experience is in software engineering and the experience is received at the company where a subject still works. It seems that either companies reward the experience at their own company more, or the experience at the company is more relevant to the tasks of that company.

The experience variable by itself is not a satisfying predictor of performance. Experience alone is only able to correctly classify 63% of the 123 (complete) cases from this study. Two other biographical variables enter into the ten variable Canonical Discriminant Function, *Total Languages Used Professionally* (which might be considered a "breadth of experience" variable), and *Math Degree Held?*, with both variables associated with exceptional performance. However, the competencies are of major importance in classifying the engineers using either the Canonical Discriminant Function or the t-test.

The competencies can be organized into four categories, *Task Accomplishment, Personal Attributes, Situational Skills,* and *Interpersonal Skills* as shown in Table 10. The competencies in each category of this table are listed in rank order based upon the mean competency score for the entire sample.

The categories shown in Table 10 form natural clusters of related competencies. *Task Accomplishment* competencies are those competencies most closely related to the unique skills or capabilities required to complete the task at hand. *Personal Attributes* are those competencies which describe inherent traits of the individual and are generally presumed to be competencies which are independent of the task itself. *Situational Skills* are the competencies that relate to the process by which an individual completes a task. *Interpersonal Skills* describe the competencies related to the interactions among the engineers.

All competencies listed are important even if they prove to not be differential between exceptional and non-exceptional performers. The list describes all of the competencies found in software engineers in this study. The list of competencies provides a well rounded view of the extent of skills, knowledge, and attributes required for a software engineer to be successful.

Five competencies are associated with exceptional performance and four competencies are associated with non-exceptional performance via the t-test. Using the ten variable discrimination function, four competencies are associated with exceptional performance (two of these are also identified with the t-test), and three competencies are associated with non-exceptional performance (two of these are also identified with the t-test. Thus, a total of seven competencies are associated with exceptional performance and five competencies are associated with non-exceptional performance.

The competencies associate with exceptional performance, *Mastery of Skills & Techniques, Maintains "big picture" View, Desire to Do/Bias for Action, Driven by a Sense of Mission, Exhibits & Articulates Strong Convictions, Pro-active Role with Management,* and *Helps Others*, generally cluster around the theme of *external focus.* The exceptional engineer is differentiated by behaviors associated with externalization — behaviors directed at people or objects outside the individual. The exceptional engineer takes a broad view of situations and develops strong convictions about how to proceed. The exceptional engineer drives toward this vision by pro-actively working with management to set goals on directions for the team. The exceptional engineer helps other engineers in an attempt to ensure the full success of the project. The one task accomplishment skill exhibited

Task Accomplishment	**T-Test**[1]	**Discrim**[2]
Leverages/Reuses Code		
Uses Methodical Problem Solving		
Mastery of Skills & Techniques	**XP**	
Writes/Automates Tests with Code		
Prior Experience		
Obtains Necessary Training/Learning		
Uses Code Reading		
Use of New Methods or Tools		
Schedules and Estimates Well		
Use of Prototypes to Asses Design		
Possesses Unique Domain Knowledge		
Uses Structured Techniques for Communication		
Personal Attributes		
Driven by Desire to Contribute	**NXP**	
Pride in Quality and Productivity		
Sense of Fun		
Lack of Ego		
Perseverance		**NXP**
Desire to Improve Things		
Pro-active/Initiator/Driver		
Maintains "big picture" View	**XP**	
Desire to Do/Bias for Action		**XP**
Thoroughness - Methodical , Organized, Cautious		
Driven by a Sense of Mission		**XP**
Exhibits & Articulates Strong Convictions	**XP**	**XP**
Mixes Personal and Work Goals		
Pro-active Role with Management	**XP**	
Situational Skills Competencies		
Concern for Reliability & Quality		
Focus on User or Customer Needs		
Thinking - Strong Analytic Skills		
Emphasizes Elegant and Simple Solutions		
Innovation		
Attention to Detail		
Design Style		
Responds to Schedule Pressure by Sacrificing Parts of the Design Process	**NXP**	**NXP**
Interpersonal Skills Competencies		
Seeks Help	**NXP**	
Team Oriented		
Helps Others	**XP**	**XP**
Willingness to Confront Others	**NXP**	**NXP**

1. Entries indicate which competencies have statistically significant differences for exceptional (marked XP) and non-exceptional (marked NXP) performers. (Based on Phase 2 Q-Sort results.)

2. Entries indicate which competencies entered the canonical discriminant function of ten variables. Competencies are marked XP (exceptional) and NXP (non-exceptional). (Based on Phase 2 Q-Sort results.)

Table 10: Competencies by Category

by exceptional engineers is *Mastery of Skills & Techniques.* This is a more self-directed competency and reinforces the fact that engineers need to be completely capable in their own discipline before they achieve the exceptional status related to an external focus. One engineer in the study states, "My perception of someone who is successful is not someone that knows the most, it is someone who can use the knowledge they do have the best."

The non-exceptional engineer is associated with five competencies, *Driven by Desire to Contribute, Perseverance, Responds to Schedule Pressure by Sacrificing Parts of the Design Process, Seeks Help,* and *Willingness to Confront Others.* Here the unifying theme is one of *internal focus.* These competencies all relate an individual acting largely alone attempting to compete tasks. The interaction with others is either one of seeking help or one of confrontation. These engineers find that they give in to the external schedule pressure and sacrifice parts of the design process that they would rather not sacrifice. The motivation of the non-exceptional engineer comes from a personal desire to contribute. This contrasts with the exceptional engineer who takes a broader view and works to influence project direction.

One way of viewing these characteristics is to place them in the context of experienced versus inexperienced individuals. Many of the competencies related with non-exceptional performance can be viewed as the behaviors of inexperienced engineers. When an engineer first begins a career, they will be unsure of their skills and capabilities. As a result, they will concentrate heavily on their own performance and exhibit an internal focus. As they mature in the job and become more confident of their skills, they will begin to take a broader view and be more pro-active in setting project direction. Thus, we find experience is a differential characteristic of exceptional performers.

The relationship between experience and competencies does not explain all of the difference in the sample, however. Many experienced software engineers never become exceptional. These experienced engineers fail to exhibit the externally focussed competencies even after many years of experience. We assume that there is a relationship between experience and certain key competencies. However, the mechanism by which experience reinforces or transfers the key competencies does not work for all individuals. Thus, we raise the question of how competencies are reinforced. Why do some software engineers use their experience to develop the competencies associated with exceptional performance while others do not? This question is beyond the scope of this research, but indicates a significant direction for future research.

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 [15, 27, 28, 29, 30].

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 [31], 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 [26]. 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 [32].

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. On the larger sample the *total*

number of languages and the *math degree held?* variables contributed to the ten variable canonical discriminant function. 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 [33, 34]. 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* [35].

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.

Littman et al also used the observational technique to study a small sample [36]. Four experienced and two novice software designers were interviewed during a two hour period while they designed an electronic mail system. They found that the experienced designers took the users view of the system before proceeding with the design. Experienced designers set goals and subgoals, used analogies to prior problems, and kept notes to monitor the progress of the design. The novice designers would plunge into the design details immediately.

The MCC and Littman et al studies provide insights into the problem solving techniques of experienced software developers; these studies did not examine the differences between exceptional and average performers. Vitalari and Dickson compared the problem-solving behavior of one low- and one high-rated systems analyst from each of nine companies [37]. Subjects verbalized their thought processes as they solved a requirements engineering problem over a two hour period. They found that the high-rated performers were more likely than low-rated performers to reject hypothesis, try several strategies, apply heuristics, set more goals, and look for analogies to prior problems. High rated performers were more likely to work for a productive relationship with the user and specify more requirements than the low-rated analysts.

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, Littman et al, and Vatalari and Dickson studies. We examine a much larger sample size than done at MCC or by Littman et al even in Phase 1. Like Vatalari and Dickson, we compare highly rated to less highly rated developers. However, we investigate the behaviors of software engineers in a larger context than one project. We study how engineers work in a team, and in an organization. None of the forgoing observational studies performed a quantitative follow-up study on a larger sample similar to our Phase 2. Thus, the significance of the results from the observational studies have not been demonstrated.

Student programmers are common subjects for studies of programmer behavior. Kagan and Douthat used extensive psychological testing to predict student performance [18]. They found a relationship between *introversion* and final success in an introductory Fortran class of 326 students. The results were based on responses to questionnaires that determine personality traits using Eysenck's Personality Inventory, the Crowne-Marlowe Social Desirability Scale, Self-Monitoring of Expressive Behavior, the Hostility Inventory, and a Type A Behavior measure.

In another study of students, Love searched for predictive factors in student programming performance [38]. The search itself is fairly brute force in that a wide array (24 factors) of data are collected for *each* run of a student assignment. Each factor is considered in an *analysis of*

variance calculation to determine predictive factors of performance. The study also attempts to relate "human information processing abilities" to programming performance. Data was collected concerning factors that affect the success of individual runs of a program, and was thus quite narrowly focused.

The extension of results from the study of students to the realm of experienced professionals is unclear at best. Since the correlation of grades and professional success is not high [39], there is no reason to expect the predictive factors from a study of students to generalize to the study of professionals.

We also used the MBTI test to determine the personality of the Phase 1 subjects. An abbreviated version of the MBTI was used by Evans and Simkin in their study of programmer productivity and demonstrated correlation between the *introversion, intuitive,* and *judging* types and performance on exams [31]. We were unable to find significant differences between the personality types of the exceptional and non-exceptional engineers.

The focus of our research is on *competencies*. A *competency* is any personal characteristic or attribute that contributes to effective performance [40]. A *job competency* is any attribute that contributes to doing a specific job well. These attributes can be specialized knowledge, an ability, an interest, a trait, or a motivation. However, they are not a job competency unless they contribute to doing the job well.

The case for studying *competencies* rather than *intelligence* was made by McClelland in a criticism of the predictive validity of intelligence tests [39]. McClelland argues that tests which sample job skills are the best predictors of competence. In order to create the tests the researcher must know which skills are necessary to achieve competent performance in a particular job. The aim of our study is to uncover these competencies.

Kelley and Caplan's development of a training program for Bell Laboratories is especially relevant [1]. They compared top performers to average workers at Bell Labs. Like our study, the top performers were those identified as "stars" by managers, but top performers also had to also be identified as stars by their peers. They found that stars do not have more innate ability than average performers. Academic talent was not a good predictor, which is consistent with our study. Nine key work strategies were identified: taking initiative, networking, self-management, teamwork effectiveness, leadership, followership, perspective, show-and-tell, and organizational savvy. Both groups agreed with these key work strategies, but their views about them differed. Networking ability and initiative accounts were described quite differently by the two groups. Taking initiative was ranked as the most important strategy by the stars, while ranked as least important by average performers. The key work strategies seem quite similar to the competencies that are identified as differential in our study. Like our study, strategies associated with externalization seem most important. Of greatist significance, Kelley and Caplan find that the skills and strategies of the stars can be taught to the average performers.

6 Conclusions

The results of our study of exceptional and non-exceptional software engineers can be summarized as follows:

- No simple predictor of performance exists.
- Experience variables are differentially related to performance, but can only correctly predict the classification of exceptional and non-exceptional performance of 63% of the subjects.
- 38 identified competencies characterize the necessary skills and attributes of professional software engineers.

- 9 competencies are differentially related to performance; they tend to cluster around personal attributes or interpersonal skills competencies.
- A discriminant function of 10 variables — 3 biographical variables and 7 competencies — can correctly predict the classification of exceptional and non-exceptional performance of 81% of the cases.

These results were obtained in a two phase investigation using professional software engineers from a major US Corporation (*The Company*). Phase 1 is a qualitative study of a small sample of software engineers that identifies the competencies. Phase 2 is a quantitative study using a larger sample that validates the the competencies and evaluates the differences between the exceptional and non-exceptional software engineers.

In Phase 1 of our research, 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, Myers Briggs Type Indicator tests, and Critical Incidence Interview data for 10 exceptional and 10 non-exceptional subjects. On the small sample used in Phase 1, one biographical factor, *Years at Company in Software*, is significantly related to exceptional performance. The Myers Briggs Type Indicator results were consistent with other studies that find most programmers exhibit the *Introvert* and *Thinking* personality types. However, using the Myers Briggs Type Indicator, we found no significant differences in personality types between the exceptional and non-exceptional software engineers on this small sample. 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. We consider the identified competencies of software engineers as *threshold* competencies — competencies that are important to the job and are exhibited equally by exceptional and non-exceptional performers.

During Phase 2 of this research, we collected data from 129 subjects. These subjects are all experienced professional software engineers engaged in the creation of software products. The data included biographical information and the results of a Q-Sort of the 38 competencies identified in Phase 1. The only biographical data demonstrating a statistically significant relationship to exceptional performance under univariate analysis are years of experience variables. In addition to years of experience, the *number of languages* and *math degree held?* variables entered into a ten variable canonical discriminant function used to classify the Phase 2 subjects. The remaining significant variable are the rankings of the competencies using the Q-Sort exercise. The analysis of the results of the Q-Sort exercise shows nine competencies are statistically related to performance under univariate analysis. Five of these competencies are more related to the behavior of exceptional performers while four of the competencies are related to non-exceptional performers. Using the multivariate technique of discriminant analysis, we find that an equation of twenty variables is able to correctly classify the exceptional and non-exceptional cases under study 86% of the time. A simplified equation of only ten variables provides correct classification 81% of the time.

The derived competencies are an important result. Even the competencies that do not differentiate between exceptional and non-exceptional performers are important. The set of competencies provides a description of the wide range of skills, knowledge, and attributes required for a software engineer to be successful.

Table 10 indicates which competencies (by category) are considered differential via the t-test and ten variable canonical discriminant function. We are led to the insight that *Personal Attributes* and *Interpersonal Skills* are most closely linked with performance differences. Skills associated with task or situation did not generally emerge as differential. This study demonstrates that the exceptional engineer can be distinguished by behaviors associated with an external focus —

behaviors directed at people or objects outside the individual. Exceptional engineers are more likely than non-exceptional engineers to maintain a "big picture", have a bias for action, be driven by a sense of mission, exhibit and articulate strong convictions, play a pro-active role with management, and help other engineers.

Acknowledgements

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. Marilyn Pultz provided invaluable assistance in applying the MBTI and in interpreting the MBTI results. We thank the anonymous US corporation (referred to as *The Company*) for allowing us access to their software engineers for this study. We are also grateful to Bob Glass who read an earlier version of this paper and provided detailed comments that greatly improved the presentation.

J. Bieman's research is partially supported by the NASA Langley Research Center, Colorado Advanced Software Institute (CASI), Storage Technology Inc., and Micro Motion 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.

References

[1] R. Kaelley and J. Caplan. How Bell Labs creates star performers. *Harvard Business Review*, 71(4):128–139, July-August 1993.

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

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

[4] G.M. Weinberg. *The Psychology of Computer Programming*. Nostrand Reinhold, New York, 1971.

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

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

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

[8] F. Brooks, Jr. *The Mythical Man-Month*. Addison-Wesley, Reading, MA, 1975.

[9] B. Shneiderman. Exploratory experiments in programmer behavior. *Int. Journal of Computer and Information Sciences*, 5(2):123–143, 1976.

[10] B. Curtis, S. B. Sheppard, and P. Milliman. Third time charm: Stronger prediction of programmer performance by software complexity metrics. *Proc. 4th Int. Conf. on Software Engineering (ICSE-4)*, pages 356–360, 1979.

[11] R. Turley. *Essential Competencies of Exceptional Professional Software Engineers*. PhD thesis, Colorado State University, Ft. Collins, CO, 1991.

[12] I.B. Myers and M.H. McCaulley. *A Guide to the Development and Use of the Myers-Briggs Type Indicator*$^{(R)}$. Consulting Psychologists Press, Palo Alto, CA, 1985.

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

[14] 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.

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

[16] Buros Institute of Mental Measurement. *The 10th Annual Mental Measurements Yearbook.* University of Nebraska, Lincoln, NE, 1989.

[17] O. Isachsen and L. Berens. *Working Together – A Personality-Centered Approach to Management.* Neworld Management Press, Coronado, CA, 1988.

[18] D.M. Kagan and J.M. Douthat. Personality and learning Fortran. *Int. Journal of Man-Machine Studies*, 22:395–402, 1985.

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

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

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

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

[23] G. McCracken. *The Long Interview,* Sage University Paper Series on Qualitative Research Methods, Vol 13. Sage Publications, Newbury Park, CA, 1988.

[24] B. McKeown and D. Thomas. *Q Methodology,* Sage University Paper Series on Quantitative Analysis in the Social Sciences, Series Number 07-066. Sage Publications, Beverly Hills, CA, 1988.

[25] W. Stephenson. *The Study of Behavior: Q-Technique and Its Methodology.* University of Chicago Press, Chicago, 1953.

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

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

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

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

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

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

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

[33] 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.

[34] 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.

[35] 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.

[36] D. Littman, K. Ehrlich, E. Soloway, and J. Black. 'You can observe a lot by just watching' how designers design. *Proc. 8th Annual Software Engineering Workshop (NASA Goddard)*, November 1983.

[37] N. Vitalari and G. Dickson. Problem solving for effective systems analysis: An experimental exploration. *Communications of the ACM*, 26(11):948–956, November 1983.

[38] L.T. Love. *Relating Individual Differences in Computer Programming Performance to Human Information Processing Abilities*. PhD thesis, University of Washington, 1977.

[39] D.C. McClelland. Testing for competence rather than for 'intelligence'. *American Psychologist*, 28(1):1–14, January 1973.

[40] Charles River Consulting. Job competence assessment: Defining the attributes of the top performer. *American Society for Training and Development Research Series*, 8, 1982.