Characterizing Software Engineering Work with Personas Based on Knowledge Worker Actions

Denae Ford
North Carolina State University
Raleigh, NC, USA
dford3@ncsu.edu

Thomas Zimmermann, Christian Bird, Nachiappan Nagappan
Microsoft Research
Redmond, WA, USA
{tzimmer, cbird, nachin}@microsoft.com

***Abstract*—Mistaking versatility for universal skills, some companies tend to categorize all software engineers the same not knowing a difference exists. For example, a company may select one of many software engineers to complete a task, later finding that the engineer's skills and style do not match those needed to successfully complete that task. This can result in delayed task completion and demonstrates that a one-size fits all concept should not apply to how software engineers work. In order to gain a comprehensive understanding of different software engineers and their working styles we interviewed 21 participants and surveyed 868 software engineers at a large software company and asked them about their work in terms of knowledge worker actions. We identify how tasks, collaboration styles, and perspectives of autonomy can significantly effect different approaches to software engineering work. To characterize differences, we describe empirically informed personas on how they work. Our defined software engineering personas include those with focused debugging abilities, engineers with an active interest in learning, experienced advisors who serve as experts in their role, and more. Our study and results serve as a resource for building products, services, and tools around these software engineering personas.**

***Index Terms*—software engineering; personas; practical knowledge work**

I. Introduction

Jobs for nonroutine cognitive work are increasing faster now than they have in the past 3 decades. In 2016, the U.S. alone reported more than 60 million workers that fit this criteria [1]. Industrial jobs that meet this criteria include finance, marketing, research, business, and software engineering. Another name for these types of employees are knowledge workers. Davenport defines knowledge workers as individuals applying their expertise to creative problems; people who think for a living [2]:

> *Knowledge workers have high degrees of expertise, education, or experience, and the primary purpose of their jobs involves the creation, distribution, or application of knowledge.*

Observing roles of knowledge work has allowed us to use a cross-industry term to describe how people work. However, we have found that even within one large software organization, each of these types of knowledge workers have different approaches to applying their expertise to creative problems.

In his work, Davenport identifies different types of knowledge workers such as controller, helper, and learner. He also describes the idea that to understand their tasks means to understand a knowledge worker [2]. In an attempt to bridge the gap between different types of knowledge workers, we start with one type, software engineers, and define how they fit under the knowledge worker umbrella.

Organizations often assume there is only one type of software engineer that fits many named job titles, but engineers disagree. Wolkov, a software engineer at LinkedIn has gone on to call out the identity crisis in software engineering,

> *"Let's go back to being an industry that hires smart engineers regardless of their title and pointing them at problems and letting them solve them without confusing them with ambiguous engineering titles [3]."*

Like many other practitioners who document this experience, Wolkov proposes we take a step back and no longer attempt to match software engineers with general job descriptions; but use practical expectations about the collaborative nature of software engineering work.

To address this concern, we construct personas based on data from software engineers on the nature of their work and how they collaborate to get their work done. We do not believe all software engineers are the same. Identifying different types of software engineers can outline their tasks and help them be more effective. Towards that end, this study aims to bridge the gap in existing research with personas based on the amount of time spent on various software engineering tasks. To the best of our knowledge, our study is the first study performed at a large software company understanding and quantifying the time spent on various software development activities in order to build personas for software engineers. Our novelty is further strengthened by the fact that we adapt Reinhardt et al. [4] knowledge worker actions to the time software engineers spend in each of those actions. We use a mixed-methods approach where we interview 21 software engineers and collect survey responses from 868 engineers to gain a unique perspective of working styles of developers in a large organization. Our results indicate a diverse set of seven personas defined on the time spent in tasks and described by their task selection process, execution, and collaboration techniques.

The major contributions of this paper are:

1) Descriptions of how software engineers collaborate with other knowledge workers on tasks at a large software company (Section V)

2) Multi-faceted personas derived from statistical analysis and qualitative descriptions of software engineering tasks (Section VI)
3) Recommendations for how practitioners can use these personas in software engineering (Section VII)

II. Personas in Software Development

In software development it is common to use personas in order to better understand users [5]. More formally, personas are behavioral specifications that embody the goals and needs of archetypical users introduced several years ago [5]. Personas were introduced to help developers understand and relate better to the users of a system. Personas are widely used in different software engineering domains ranging from requirements engineering [6] to tool development [7] and security and privacy [8].

Software companies use a variety of personas in varying stages of their development cycle though vary rarely are they supported by data. One example of this is when Microsoft used the personas Mort, Elvis, and Einstein to define developer profiles for tool features [9].

Personas can also be used to understand the overall usability to aid early in the design process. As stated by Pruitt and Grudin [10], personas can provide, "a broad range of qualitative and quantitative data, and focus attention on aspects of design and use that other methods do not." Pruitt and Grudin found that once personas are generated, it is easier to craft them into new situations rather than recreate a user scenario. Identifying users provides more information on how they can interact with a system.

In addition, Freiss found that though persona descriptions may not be explicit, they can help keep the design direction user-focused [11]. One example of personas that follow this pattern is GenderMag. GenderMag uses concrete facets of gender differences, such as processing style and motivations, to describe ways to encourage inclusive software development [12]. Though genders were not concrete identifiers of each declared personas, Marsden et al. performed experiments that confirm perceptions of persona attributes with particular genders [13]. This evaluation using GenderMag personas is not only an example of how helpful personas can be in explaining concepts, but also a building block for further experiments to confirm how personas can be adapted.

III. Research Questions

We frame our study in the context of how software engineers work. Our approach for determining how software engineers work is grounded in knowledge worker actions as described by Reinhardt and colleagues [4]. This study is guided by three research questions.

RQ1 How do software engineers spend their time in knowledge worker actions?

Previous work defines distinction between knowledge workers based on the actions taken to complete a task. Reinhardt et al. surveyed individuals at a large organization about their tasks [4]. In this work, Reinhardt et al. identified a typology

TABLE I
Adapted Knowledge Worker Actions for Software Engineers

Learning	Learning and acquiring new knowledge, skills, or understanding, for example taking courses, training, or observing others
Analyze	Examining something carefully in order to understand it, for example, understanding unfamiliar code, debugging, reviewing log output
Authoring	Independently creating text or media content NOT with other people, for example, writing code for personal projects, documentation, tutorials, architecture documents, creating technical presentations, etc.
Co-authoring	Collaboratively creating text or media content with other people, for example, writing code with a team (such as company projects), documentation, tutorials, architecture documents, creating technical presentations, etc.
Dissemination	Sharing information about work results, for example, presenting in technical meetings or brown bag lunches
Expert Search	Finding an expert to help discuss or solve a problem, for example finding someone to help fix a bug
Feedback	Providing feedback on technical documents and code, for example, code reviews, or architecture reviews
Information Organization	Managing personal or organizational information, for example, taking notes, tracking and prioritizing work for yourself or others
Information Search	Retrieving specific information, for example, finding a bug report, wiki page, specification, or API documentation
Monitoring	Keeping oneself or the organization up-to-date about selected topics, for example, public tasks lists, reading status reports
Networking	Interacting with other people and organizations to exchange information and develop contacts, for example, attending meet-ups, conferences, participating in online discussions
Service Search	Finding specialized tools, for example tools to visualize data

of how knowledge worker actions further define the tasks execution processes of knowledge workers. Reinhardt et al. also present each knowledge worker action in terms of guiding roles. We adapt knowledge worker actions to tasks as they apply to what software engineers encounter in their domain as shown in Table I. We keep Reinhardt et al. knowledge worker actions names to support qualitative research transferability.

RQ2 How do software engineers decide on their task, task inputs, and task outputs?

Davenport describes a factor to understand knowledge workers through complexity of work based on routine-repeatable work and judgment-improvisational work [14]. There are common tasks for which systematic and highly reliant on formal rules and procedures are required and some that do not. To apply this to software engineers, we are interested in the range of autonomy for software engineers to select tasks, determine task that need to be done based on ongoing work and deciding when to move on to the next task. With these dynamics supported by software engineering literature [15], [16], we identify these as valuable details to determine distinctions between software engineers.

RQ3 How do software engineers collaborate to complete a task?

Software development at a large software company is based on individuals working on different parts of a project and then being able to join individual pieces together for a successful final product. Davenport referenced collaboration and the interdependence of tasks as one way to identify differences between knowledge workers [14].

IV. Methodology

To answer our research questions, we used a mixed-methods approach of semi-structured interviews, to qualitatively understand tasks software engineers complete, and surveys, to gather quantitative insight on how time is spent in knowledge worker actions. All study materials can be found in our online appendix [17].

IV.A. Semi-Structured Interviews

Protocol. We organized interviews to better understand how software engineers explain their work. We conducted half hour, semi-structured interviews with software engineers at a large software company. Each interview was led by the first author, who was often accompanied by the second or third author. All interviews were held in person or via Skype. Interviews were audio recorded and later transcribed for analysis. Each interview consisted of four parts, motivated by previous research on knowledge workers.

1) *Role of the participant*. We asked about the role of the participant at the company and what types of projects they work on.
2) *Tasks, inputs, and outputs*. These questions were motivated from Davenport's discussion [14] on how knowledge workers are defined by the tasks they manage. We asked participants for three of their most common tasks completed in their role and three tasks that might be uncommon, but important to complete. Motivated by Thomas and Baron [18], we asked about the inputs and outputs of these tasks to better understand productivity of software engineers in terms of knowledge work. For inputs, we asked who selects the tasks and how. For outputs, we asked who receives outcomes of a task and how they are used. We also asked participants about specific tasks from the previous work day to help ground their responses.
3) *Knowledge worker actions*. We asked participants how often they perform knowledge worker actions introduced by Reinhardt and colleagues [4]. We also asked who they perform these actions with.
4) *Collaboration*. We asked participants about how they collaborate with other colleagues. Specifically, we were interested in the roles of their collaborators and whether or not they were a member of their team. We also asked about any tools they may use to facilitate collaboration on these tasks.

Participants. We randomly selected 99 employees with *Software Engineer* listed as their job title. We sent each an invitation email and described our interests in their roles, tasks, and projects to characterize software engineers as knowledge workers. We received responses and interviewed 21 participants. Participants were compensated for their time at the end of each interview with a ten dollar meal voucher.

Data Analysis. We used a transcription service to transcribe the audio recorded from interviews. We then coded interview responses to questions about knowledge worker actions and included responses to their time spent, task execution process, and task selection in the survey. For example, participants emphasized how often they collaborate with individuals in similar roles, but on different teams. We also included this in our survey as a free-response question.

IV.B. Follow-Up Survey

Protocol. We created a survey to gather and validate a wide range of responses on how software engineers spend their time. On average, participants took 15 minutes to complete the survey. Our survey consisted of 30 questions as shown in Table II. We used the guidelines for personal opinion surveys by Kitchenham and Pfleeger [19] and guidelines from Harvard Business Reviews for workplace surveys [20] to design our survey.

Pilot Survey. We first conducted a pilot survey to capture a range of tasks and collaboration styles that may not have been gathered in interviews. The pilot survey included questions that were asked in the interview including time spent, task execution, and task selection. We used a combination of responses from the interviews and pilot survey to inform multiple choice options on the final survey about tasks and execution strategies that software engineers have. All questions regarding tasks were left as open ended questions.

The questions in the survey fell into the following groups.

1) *Demographics.* We collected demographic information such as current role, years of experience, gender, and internal organization. Additional questions included types of projects respondents worked on, novelty of projects, and ability to make independent decisions for projects.
2) *Tasks.* We asked about the inputs and outputs that are common to software engineering tasks, who selects a task, and when a task is completed. We offered survey respondents the opportunity to write in an answer for questions if the options listed did not apply to their tasks.
3) *Time spent.* Inspired by Reinhardt and colleagues, we included a list of knowledge worker actions in the context of software engineers and asked respondents to estimate the number of hours they spend on each action a week. We adapted knowledge worker actions to software engineering (Table I) from interview responses about daily routines.
4) *Collaborations.* We asked respondents to select the job titles of people who they regularly interact with and

TABLE II
OVERVIEW OF SURVEY QUESTIONS

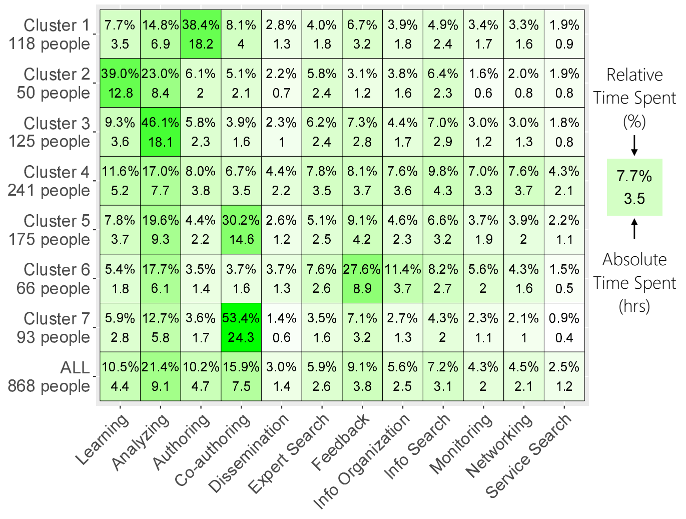
DEMOGRAPHICS
Questions about → experience, gender, team size, other demographics (works on open source, internal projects, autonomy, decision maker, expert), self perceptions (thinker, doer, innovator, planner, etc.)

TIME SPENT
Please enter roughly how many hours per week you typically spend on each of the activities. → the list of 11 knowledge worker actions adapted from Reinhardt and colleagues [4] instantiated to the software engineering domain

How many hours in a week are spent → debugging code, writing code/creating text/media for personal projects, writing code/creating text/media collaboratively, in code reviews, testing code, with emails, in meetings?

TASKS
What are the inputs for your various tasks?
What are the outputs for your various tasks?
Who selects the tasks you work on?
How do you know you are done working on a task?

COLLABORATIONS
Select the job titles of the people you regularly interact within your team, outside your team, or both.
How many reviewers are included in code reviews of one of your changes?
Have you ever been brought on a different team to put out fires?



	Learning	Analyzing	Authoring	Co-authoring	Dissemination	Expert Search	Feedback	Info Organization	Info Search	Monitoring	Networking	Service Search
Cluster 1 118 people	7.7% 3.5	14.8% 6.9	38.4% 18.2	8.1% 4	2.8% 1.3	4.0% 1.8	6.7% 3.2	3.9% 1.8	4.9% 2.4	3.4% 1.7	3.3% 1.6	1.9% 0.9
Cluster 2 50 people	39.0% 12.8	23.0% 8.4	6.1% 2	5.1% 2.1	2.2% 0.7	5.8% 2.4	3.1% 1.2	3.8% 1.6	6.4% 2.3	1.6% 0.6	2.0% 0.8	1.9% 0.8
Cluster 3 125 people	9.3% 3.6	46.1% 18.1	5.8% 2.3	3.9% 1.6	2.3% 1	6.2% 2.4	7.3% 2.8	4.4% 1.7	7.0% 2.9	3.0% 1.2	3.0% 1.3	1.8% 0.8
Cluster 4 241 people	11.6% 5.2	17.0% 7.7	8.0% 3.8	6.7% 3.5	4.4% 2.2	7.8% 3.5	8.1% 3.7	7.6% 3.6	9.8% 4.3	7.0% 3.3	7.6% 3.7	4.3% 2.1
Cluster 5 175 people	7.8% 3.7	19.6% 9.3	4.4% 2.2	30.2% 14.6	2.6% 1.2	5.1% 2.5	9.1% 4.2	4.6% 2.3	6.6% 3.2	3.7% 1.9	3.9% 2	2.2% 1.1
Cluster 6 66 people	5.4% 1.8	17.7% 6.1	3.5% 1.4	3.7% 1.6	3.7% 1.3	7.6% 2.6	27.6% 8.9	11.4% 3.7	8.2% 2.7	5.6% 2	4.3% 1.6	1.5% 0.5
Cluster 7 93 people	5.9% 2.8	12.7% 5.8	3.6% 1.7	53.4% 24.3	1.4% 0.6	3.5% 1.6	7.1% 3.2	2.7% 1.3	4.3% 2	2.3% 1.1	2.1% 1	0.9% 0.4
ALL 868 people	10.5% 4.4	21.4% 9.1	10.2% 4.7	15.9% 7.5	3.0% 1.4	5.9% 2.6	9.1% 3.8	5.6% 2.5	7.2% 3.1	4.3% 2	4.5% 2.1	2.5% 1.2

Fig. 1. The seven clusters of how software engineers spend their time. Each row corresponds to a cluster and each column to a knowledge worker action. The top number in each cell indicates the average relative time spent (as a fraction) and the bottom number the absolute time spent (in hours). The average relative time spent for each cluster sums to 1.

identify whether the collaboration with that role is within their team, outside their team, or a combination of both. We also asked about code reviews and whether respondents have been brought on teams to help with emergencies as we found this to be a common event participants mentioned in interviews.

Participants. We invited 4594 employees with *Software Engineer* listed as their job title at a large software company and received responses from 891 respondents (response rate 19.4%). We offered respondents the option to enter a raffle for four $75 Amazon.com gift certificates. The median years of professional experience as a software engineer is 7 years and the median years of reported experience at their current company is 3 years.

Data Analysis. We analyzed the survey data in three phases. First, we applied clustering on the time spent on knowledge worker actions and resulted in seven clusters. Next, in the descriptive analysis, we used the responses from the entire survey to characterize each cluster. In the final phase, we used the cluster descriptions to develop personas.

Clusters → Cluster Descriptions → Personas

Clustering. To answer **RQ1**, we used the responses to the number of hours a week listed for each knowledge worker action to cluster the survey participants in three steps:

Step 1: We included only participants who responded to the time spent question, i.e., the sum of time spent in all knowledge worker action was greater than zero hours. We only wanted to include participants that explained how their time was spent. Of the 891 survey respondents, 868 entered hours for each knowledge worker section.

Step 2: We normalized the time each person spends across knowledge worker actions by computing the percentage a person spends in each action.

Step 3: We used k-means clustering [21] to iteratively group respondents who reported similar relative time spent for each knowledge worker action into groups. To determine the appropriate number of clusters we plotted the within groups sum of squares by the number of clusters and identified a bend in the plot. We observed the bend at $k = 7$ and manually inspected the clusterings for 7 to 10 clusters. We chose the clustering with 7 clusters because the cluster sizes were more balanced, i.e., no cluster represented only a small set of people.

The result of the clustering is shown in the heat map in Figure 1. The top row corresponds to the population of 868 developers who responded to the time spent question. Each subsequent row corresponds to a cluster. For example, the second row corresponds to Cluster 1 which represents 118 people. Each column corresponds to a knowledge worker action. A cell (x, y) indicates the average time that developers in a cluster x spend on action y. For example, developers in Cluster 1 spend on average 7.7% of their time on Learning; the 7.7% correspond to 3.5 hours.

Cluster Descriptions. To answer **RQ2**, we used statistical differences between tasks, task inputs, and task outputs reported as cluster descriptions. The results are summarized in Table III. We used a decision tree to explain how the clustering algorithm built the clusters and to characterize the clusters with respect to the time spent question. The results are in the left column of Table III. For example, most developers is Cluster 3 spent less than 19.29% *Co-authoring* and at least 31.41% *Analyzing*. In Cluster 7, most developers spent at least 42.18% of their time *Co-authoring*.

For each cluster, we identified statistically significant differences with respect to how developers responded to the other questions in the survey. We used the *Fisher Exact Value* (for binary variables) and *Mann-Whitney U* tests to determine significant differences. The results of these test are summarized in the right column of Table III. For example, developers in Cluster 2 have significantly more entry-level engineers than the other clusters (52.0% vs. 30.6%). Software engineers in this cluster also have less professional experience than other clusters (5.2 vs. 8.8 years).

Persona Development. In the final phase, we used the cluster descriptions and interview transcripts to infer personas that are representative of each cluster. We use a narrative style to present the personas in Section VI, which is recommended in user experience design to describe personas [22].

In addition, after personas were identified, we conducted member checking [23] with interview participants to confirm personas they identify with. We sent interview participants a survey, similar to that described in Section IV-B and received responses from 20 of the 21 interview participants. We were not able to reach one participant who left the company. We determined their time spent distributions based on their responses to interview questions about how they spend a typical day. We were able to identify each interview participant with a persona.

V. Collaboration Styles

To answer **RQ3**, we explored collaboration in two primary ways. First, we investigated *who* software engineers collaborate with in terms of different roles in their own and other teams. Second, we explored *how* they collaborate by examining the methods of collaboration and channels of communication that the engineers employ.

During the interviews we asked participants about who they conduct knowledge worker actions with and how often. We also asked about who the software engineers collaborate with in our survey. When asked about the roles and how often they collaborate with other people on their teams, participants mentioned how collaborations occurred within and outside their team; acknowledging that they also exist across different roles.

Table IV shows a summary of the roles that software engineers collaborate with within and outside of their teams (and both), based on survey responses. Unsurprisingly, software engineers collaborate mostly with other software engineers, both on their team and other teams. The non-engineer roles that they collaborate highly with are their engineering managers and project managers. Finally, the roles outside of the teams that they collaborate with most are scientists (e.g., data scientists and applied scientists) and operations specialist (e.g., business operations and service operations). When we quantitatively examined the roles that software engineers collaborate with across the seven clusters the only statistically significant difference we found was that Cluster 5 collaborates with operations specialists about half as much as the average. We thus conclude that there is little difference in *who* each of the clusters collaborates with.

A number of forms of collaboration emerged during our interviews. Reviewing code and co-authoring code were the primary collaborative activities, while the primary collaborative communication channels were meetings, email, and impromptu office discussions (often mentioned in interviews as interruptions).

V.A. Collaborative Cluster Comparison

Although we found little difference in the roles that clusters collaborate with, we discovered that the types and frequency of collaborative activities were some of the most differentiating aspects of these clusters.

For example, Cluster 7 is the one of least collaborative clusters, as they spend less time in meetings, writing email, or dealing with interruptions from other engineers. Their outputs are engineering artifacts (code changes and bug reports) rather than items intended to be shared with others (reports, presentations, and analysis results). The only collaborative activity that they engage in highly is co-authoring code for the company, something they do four times as much as the other clusters. Cluster 2, comprised of more entry level engineers than any other cluster, is similar in that they are minimally collaborative on nearly all fronts; the difference between these clusters being that Cluster 2 spends less time co-authoring code while Cluster 7 spends dramatically more.

In contrast, Cluster 6, is highly collaborative. Engineers in this cluster spend far less time writing code compared to the average, but spend around twice as much time conducting code reviews, in meetings, writing emails, and answering interruptions. They are often experts (which enables them to provide useful feedback) and are twice as likely to be called in to put out fires. Both their inputs and outputs are also highly collaborative, including handling customer concerns, management inquiries, ad-hoc team discussions, as well as making presentations and writing design documents.

Other clusters are more nuanced in their forms of collaboration. For instance, Cluster 4 spends more time in meetings and writing email than the average, but participates less in code reviews and co-authoring code, while Cluster 5 spends more time providing feedback in code reviews than the average and dramatically more time co-authoring code.

V.B. Code Review

Reviewing code that goes into a major product is an important part of the development process [24]. In interviews participants acknowledged code review as one of the most collaborative aspects of their job. The code review process involved multiple knowledge worker actions. In the code review process, the authoring software engineer must find the most appropriate engineers to review their changes (*Expert Search*) and the reviewing software engineers offer suggestions on how to enhance the code (*Feedback*). From the survey, participants indicated on average they spend approximately 3.2 hours in code review per week and that the typical code review involves four people.

Depending on the particular task, writing code can be a collaborative activity [25]. Engineers working on one

TABLE III
DESCRIPTIONS OF SOFTWARE ENGINEERING CLUSTERS

Cluster	**Significant differences with respect to demographics, time spent, tasks, inputs, and outputs**			
Cluster 1	↑ Professional experience	10.0 vs. 8.4 years	↑ Output: source code documentation	56.4% vs. 44.7%
Co-authoring $< 19.29\%$	↑ Work on internal projects	53.1% vs. 41.6%	↑ Output: detailed team reports	40.4% vs. 26.6%
Analyzing $< 31.41\%$	↑ Have a lot of autonomy	65.5% vs. 54.7%		
Authoring $\geq 23.67\%$	↓ Time: debugging code	4.1 vs. 5.5 hours		
OR	↑ Time: writing code for personal projects	5.5 vs. 1.3 hours		
Co-authoring $\geq 19.29\%$	↓ Time: writing code for company	3.3 vs. 7.8 hours		
Co-authoring $< 42.18\%$	↓ Time: code reviews	2.6 vs. 3.3 hours		
Authoring $\geq 19.18\%$				
Cluster 2	↑ Entry-level engineer	52.0% vs. 30.6%	↓ Input: customer concerns	30.3% vs. 51.6%
Co-authoring $< 19.29\%$	↓ Professional experience	5.2 vs. 8.8 years	↓ Input: ad-hoc discussions with team	30.3% vs. 69.0%
Analyzing $< 31.41\%$	↓ Company experience	2.9 vs. 5.0 years	↓ Output: data analysis result	15.6% vs. 37.6%
Authoring $< 23.67\%$	↓ Identify as expert on a team	12.8% vs. 52.3%	↓ Output: product feature	43.8% vs. 62.7%
Feedback $< 19.29\%$	↓ Have a lot of autonomy	27.7% vs. 58.0%	↓ Tasks: I chose tasks	45.5% vs. 64.0%
Learning $\geq 23.75\%$	↓ Flexible to make decisions on team	34.0% vs. 58.1%	↓ Tasks: program manager selects task	12.1% vs. 29.0%
	↓ Time: writing code for company	3.1 vs. 7.3 hours		
	↓ Time: code reviews	1.4 vs. 3.3 hours		
	↓ Time: being interrupted	4.2 vs. 5.6 hours		
Cluster 3	↓ Work on internal projects	31.8% vs. 44.9%	↓ Input: user scenarios	59.2% vs. 72.2%
Co-authoring $< 19.29\%$	↓ Identify as expert on a team	33.6% vs. 52.5%	↓ Input: customer concerns	40.8% vs. 52.2%
Analyzing $\geq 31.41\%$	↑ Changed from expert to non-expert	54.2% vs. 38.1%	↓ Input: ad-hoc discussions with team	57.3% vs. 68.9%
	↓ Degree of independence	4.3 vs. 5.0 points	↓ Output: source code documentation	35.3% vs. 48.1%
	↑ Time: debugging code	10.3 vs. 4.5 hours	↓ Output: presentations	20.6% vs. 33.2%
	↓ Time: writing code for company	1.5 vs. 8.0 hours	↓ Output: design documents	32.4% vs. 46.1%
	↓ Time: meetings	4.4 vs. 5.1 hours	↓ Output: product feature	52.9% vs. 63.3%
	↓ Input: specifications	46.6% vs. 60.0%	↓ Output: detailed team reports	17.6% vs. 30.2%
	↓ Input: business requirements	34.0% vs. 54.9%	↑ Tasks: done when formal agreement from peer developer	13.0% vs. 5.9%
Cluster 4	↑ Entry-level engineer	38.6% vs. 29.3%	↓ Output: source code check-ins	89.7% vs. 96.7%
Co-authoring $< 19.29\%$	↓ Time: debugging code	4.2 vs. 5.8 hours	↓ Output: bug fixes	86.2% vs. 94.5%
Analyzing $< 31.41\%$	↓ Time: writing code for personal projects	1.8 vs. 1.9 hours	↑ Output: presentations	38.5% vs. 28.7%
Authoring $< 23.67\%$	↓ Time: writing code for company	2.4 vs. 9.0 hours	↑ Output: detailed team reports	36.4% vs. 25.4%
Feedback $< 19.29\%$	↓ Time: code review	2.9 vs. 3.4 hours	↑ Tasks: program manager selects task	35.0% vs. 25.6%
Learning $< 23.75\%$	↑ Time: emails	6.3 vs. 5.6 hours	↓ Tasks: done when code checked into repository	39.4% vs. 51.4%
	↑ Time: meetings	5.7 vs. 4.8 hours		
	↓ Input: code from other engineers	61.2% vs. 72.5%		
	↓ Input: live issues they noticed	51.0% vs. 60.0%		
Cluster 5	↓ Collaborate with Operations Specialist inside their team	18.9% vs. 40.0%	↑ Input: management inquiries	60.7% vs. 50.4%
Co-authoring $\geq 19.29\%$	↓ Time: writing code for personal projects	0.8 vs. 2.2 hours	↑ Input: code from other engineers	80.0% vs. 66.6%
Co-authoring $< 42.18\%$	↑ Time: writing code for company	12.3 vs. 5.6 hours	↑ Input: ad-hoc discussions with team	75.2% vs. 65.2%
Authoring $< 19.18\%$	↑ Time: code review	3.6 vs. 3.1 hours		
Cluster 6	↓ Entry-level engineer	12.1% vs. 33.5%	↑ Input: design documents	70.6% vs. 55.4%
Co-authoring $< 19.29\%$	↑ Professional experience	10.7 vs. 8.4 years	↑ Input: business requirements	66.7% vs. 50.7%
Analyzing $< 31.41\%$	↑ Company experience	6.9 vs. 4.7 years	↑ Input: user scenarios	88.2% vs. 68.9%
Authoring $< 23.67\%$	↓ Work on open source	1.6% vs. 9.9%	↑ Input: customer concerns	70.6% vs. 49.0%
Feedback $\geq 19.29\%$	↑ Identify as expert on a team	73.4% vs. 48.1%	↑ Input: management inquiries	66.7% vs. 51.4%
	↓ Work on not yet release projects	23.4% vs. 38.1%	↓ Input: API documentation	39.2% vs. 54.5%
	↑ Put out fires	82.0% vs. 40.4%	↑ Input: ad-hoc discussions with team	80.4% vs. 66.2%
	↓ Time: debugging code	3.8 vs. 5.5 hours	↑ Output: presentations	47.1% vs. 30.2%
	↓ Time: writing code for personal projects	0.7 vs. 2.0 hours	↑ Output: design documents	66.7% vs. 42.4%
	↓ Time: writing code for company	1.6 vs. 7.5 hours	↑ Tasks: I chose tasks	78.4% vs. 61.9%
	↑ Time: code review	7.0 vs. 2.9 hours	↑ Tasks: product owner selects task	21.6% vs. 7.6%
	↑ Time: emails	9.4 vs. 5.5 hours	↓ Tasks: done when code approved by code review	29.4% vs. 45.9%
	↑ Time: meetings	7.6 vs. 4.8 hours		
	↑ Time: being interrupted	9.4 vs. 5.3 hours		
Cluster 7	↓ Time: debugging code	4.0 vs. 5.5 hours	↑ Output: source code check-ins	100.0% vs. 94.0%
Co-authoring $\geq 42.18\%$	↑ Time: writing code for company	20.9 vs. 5.0 hours	↑ Output: bug fixes	98.8% vs. 91.3%
	↓ Time: emails	4.0 vs. 6.0 hours	↓ Output: presentations	22.1% vs. 32.7%
	↓ Time: meetings	3.7 vs. 5.2 hours	↓ Output: data analysis result	26.7% vs. 38.0%
	↓ Time: being interrupted	4.0 vs. 5.8 hours	↓ Output: detailed team reports	12.8% vs. 30.6%
	↑ Input: bug reports	84.9% vs. 74.3%	↑ Tasks: done when deliverables are tested	73.3% vs. 59.7%

TABLE IV
Breakdown of Roles Software Engineers Collaborate Within/Within & Outside/Outside Their Team

	Within	**Within & Outside**	**Outside**
Project Manager	41.80%	42.70%	15.50%
Software Engineer	41.30%	55.20%	3.60%
Software Engineer 2	37.30%	59.40%	3.20%
Senior Software Engineer	33.00%	63.00%	3.90%
Principal Software Engineer	29.70%	54.70%	15.60%
Software Engineering Lead	40.00%	53.80%	6.20%
Engineering Manager	53.20%	40.90%	6.00%
Architect	31.70%	29.50%	38.80%
Designer	38.50%	18.00%	43.40%
Scientist	23.60%	14.60%	61.80%
Operations Specialist	16.00%	63.70%	20.30%

component often must consult with owners of related or dependent components (*Expert Search*). Interview participants often sought out expertise and ideas from team members prior to undertaking a coding task. Some teams practice pair programming, which is a nearly constant collaborative process. We therefore asked survey respondents how much time they spend writing code for the company (presumably in collaboration with other company engineers); the average time reported was 9.4 hours.

V.C. Time Spent in Meetings

Prior work acknowledges many organizational tasks that as a collaborative part of the software engineering workflow [26]. These tasks require some method of communication between a software engineer and other team members. Thus, we also asked participants about the amount of time participants spend in meetings, emails, and interruptions. Participants revealed that they spent approximately 5 hours of their time in meetings, 6 hours in emails, and 6 hours of their time being generally interrupted on average over the course of a week.

VI. Software Engineering Personas

To interpret our findings from **RQ1**, **RQ2**, and **RQ3**, we give a face to clusters through seven personas to describe software engineers: Acantha, Lilo, Isabelle, Cameron, Iman, Ava, and Ciara. We used time spent clusters and cluster descriptions to inform our personas. In addition, we support each persona with quotes from an interview participant, labeled S#, who identified with the persona. We selected quotes from interview participants that demonstrate depth of distinct personas.

VI.A. Acantha, the Autonomist Acumen ← Cluster 1

Acantha is known for her autonomy and keen abilities to make good judgments when writing code. Her long tenure as a software engineer has supported her decision making abilities. She spends the majority of her time writing code independently on personal projects and also spends a significant amount of time working on internal projects. On average, more than 20% of her time is spent in the *Authoring* knowledge worker action.

Interview participants who identified with this persona also talked about how their professional experience has affected their success. S8 mentions how their autonomy and experience have greatly influenced their risk taking abilities:

> *You know, I'm very much a risk-taker and I'd much rather ask for forgiveness. I'd much rather cross the line and wait for somebody to catch me. I'm willing to cross the line, and I do, and occasionally I get my hand slapped. I'm in this wonderful position.*

VI.B. Lilo, the Continuous Learner ← Cluster 2

Lilo is known for her interest in learning and getting adapted to a new team early in her career. She spends, on average, more than 23% of her time in the *Learning* knowledge worker action. She takes a great interest in watching online lectures to brush up on the latest frameworks and technologies that she may be able to add to a new project. At this stage in her career she is often assigned tasks rather than choosing tasks to work on. Though she may be an entry-level engineer with little professional experience at the large company, she takes an initiative to get caught up quickly on team projects.

Interview participants who identified with this persona reflected on being relatively new to their organization. The individuals who identified with this persona also do not claim to be an expert on the team. S7 talks about how their lack of experience encourage them to find someone who did:

> *I mean it's usually scoped but we don't know the A, B, C's of it, like where to start, how to start and any such things. But it usually starts with talking to different people. We know at least one contact who would then redirect us to multiple other contacts or they redirect us to some other people.*

VI.C. Isabelle, the Investigator ← Cluster 3

Isabelle is best known for being the top investigator on her team. She spends the most time, over 30% on average, in the *Analyzing* knowledge worker action. She spends much time in the trenches of projects debugging and making sense of code. Isabelle recently changed from being one of the top experts on her team to being a novice in her new team. However, this has not impeded her success as she was able transition smoothly and adapt the same techniques of connecting with peer developers to ensure success on her new team.

Interview participants who identified with this persona also spent a significant amount of time debugging code. S1 describes the process for debugging:

> *It depends on what kind of task. For example, to fix bugs in theory you need to reproduce the problem and to just do a debug step-by-step, going into the very developed source code to figure it out. That takes a long, long time sometimes.*

VI.D. Cameron, the Communicator ← Cluster 4

Cameron is considered to be a communicator because of her ability to distribute her time evenly and early in her career while securing the logistics of her work environment. She is able to spread her time across all knowledge worker actions well. However, since she is so early in her career she also seeks guidance from her program manager and others on how to approach tasks. Cameron spends a lot of time communicating about how to set up and approach her work through emails, meetings, reports, and presentations, but that sometimes gets in the way of writing code.

Interesting enough, interview participants who identified with this persona also facilitated a lot of knowledge sharing, which can inhibit writing code. S5 talks about not being able to produce code, but arranging resources:

> *The concentrated work time usually involves coding. Right now it involves learning, collating documentation, arranging resources for the people who are gonna come after me.*

VI.E. Iman, the Interactive ← Cluster 5

Iman is known for her activity with collaborative code writing style but also for her dynamic level of interaction that goes on within her team in terms of the *Co-authoring* knowledge worker action. Aside from being very helpful in code reviews, much of her tasks to complete come from everyone else. On average, many of her tasks come from management questions, source code from other engineers, and also spontaneous conversations within her team.

Interview participants who identified with this cluster also put a high value on interacting with team members and refining tasks with others. S19 talks about how her project stages encourage these ad hoc conversations:

> *So we do all the code review ourselves - that's a very important part. [Another] is the stand-ups, which have to be done - not every day, but twice in a week, we have meetings. And that's very important because as we are still in a preview tool, so we have a lot of discussions.*

VI.F. Ava, the Advisor ← Cluster 6

Ava is known for the high value she places on being helpful to others and giving feedback. On average, she spends over 20% of her time offering feedback. She has significantly more professional experience, identifies as an expert, and puts out more fires than others when her team is in need of an emergency product fix. Her professional experience shines through the most as she has more years at her large company than others. Her ability to put out fires and help the team in unplanned emergency situations demonstrates how often she gets interrupted while working.

Interview participants who also identified with this persona place a high value on advising and helping others. S4 mentioned trying to help people put out their own fires by pointing them in the right direction when they come to them for help:

> *That's where people have an actual coding problem and they don't know what to do to proceed, and you're trying to give them advice or "here's an example of code you can follow," or "go read this article." You're kind of trying to hand them enough information so they can go solve the problem.*

VI.G. Ciara, the Team Coder ← Cluster 7

Ciara embodies this idea of being a team coder for her immense amount of time in the *Co-authoring* knowledge worker action (sometimes over 40%). She spends a significant amount of time producing code for her large company with a team by her side. Her major tasks are to sift through bug reports and fix them with source code check-ins. Ciara does not spend a lot of time in emails, meetings, or presentations because she is busy testing bug fixes.

Interview participants who identified with this persona also emphasized that they *do bug fixes often*. S21 discusses one main focus of their role includes testing fixes:

> *My role, in particular has three things that we focus on. One is basically our automated test runs. This helps us keep the business running. Basically, when the service team deploys changes or updates stuff or new content comes in, we have a suite of tests that run four times a day. So a lot of times something might roll out and because they're the service team, they don't really do as much robust testing from the client side.*

VII. Discussion

VII.A. Persona Takeaways

Inclusion through Persona Use. Ultimately the goal of identifying personas is to allow user experience researchers to help users with products, services, or navigate a landscape. However, how can we *effectively* help users navigate this space if we do not know who those users are?

Personas inject a human aspect that is often missed when discussing those who make software. Thinking about different cultural approaches to knowledge work from a demographic, geographic, team, and tenure perspective encourages the idea that *people* make software and these same people work based of their prior experiences. For example, Burnett et al. [12] include a range of learning styles that inform widely used and insightful personas that can be connected to real people. Identifying these personas encourages researchers and practitioners to think broadly of all the users and helps us be more inclusive of the many types of users. This will also encourage others to plan for the different types of software engineers that have yet to be identified. In addition, this will help include a wide array of perspectives in the dialogue for how software engineers work. Thus, opening the opportunity for an inclusive invitation for others, such as women, to join this field.

Transient Modes. With these personas, we want to make clear that these personas identify some approaches to how software engineers work and note there is more work to be done. One example is transitions between personas. Participants from interviews and open responses in the survey mentioned this idea of having to switch between modes on a project. Some emphasized that it depends on what project has demanding deadlines at that particular time. We have seen this same dependency on deadlines and task distribution in other knowledge worker spaces [27].

In support of transitions, interview participants mentioned how they worked on a team until becoming an expert. After being considered as one of the experts on the team, the participant switched teams and were then considered a novice to that new team but were able to adapt swiftly. This phenomenon would be interesting to study at scale and determine what exactly are those particular skills that aid in re-training to become an expert in another team. Two question here are, a) *What strategies have prior experts developed that facilitate their smooth transition to the next team?* and b) *How is this transition demonstrated across personas?*

Additional Personas. Identifying these personas now allows us to be more aware of the different types of personas. However, this also opens up the idea that there could be more. Some of the clusters we determined have many more software engineers than others; some with as much as a 100 employee difference. This demonstrates the idea that if we expanded the cluster with a larger range we may have determined many more clusters across time spent in knowledge worker actions. In addition to the hours of time spent across knowledge worker actions, the significant differences across the self-reported descriptions of autonomy and the task execution process allude to the possibility that there are many other dimensions that influence what a software engineer does, for example tools and technology used by engineers.

VII.B. Implications

Research. This work offers much insight to a new direction for conducting software engineering studies. We encourage empirical software engineering researchers to report more details about the persona of the software engineers they may build tools for or conduct studies with. Now that we have much literature of how software engineers approach their tasks we can now design studies that cater to each of these personas.

We can question whether there are certain features of development that attract one persona more than another. For example, *Should we build tools that tailor to Ciara's collaborative nature of writing code with a team or build tools that can address multiple styles of collaboration?* Another example of this is if a software engineer identified best with Isabelle, they may have a greater interest in advanced features of debugging tools that can facilitate an analysis. These personas can help companies understand tradeoffs of supporting one persona over another.

Practice. With this work, software engineering teams in industry can now cater to these roles and have company courses to enhance their employee experience according to their persona description. We are not implying that a software engineer will always fit into one persona and have an interest that will always relate to that persona. However, we are recommending that technology organizations encourage software engineers to seek resources to support the working style of software engineers on these types of projects.

For example, if a software engineer identifies best with Lilo, then software engineers should have company resources that will encourage them to stay informed of new technology as they continue to work on new projects to help this individual be effective.

VIII. Limitations

One major constraint of this work is that we only sampled one large company. As companies and organizations have different cultures, it may not be clear how this work may be able generalize to online communities, for example. However, Microsoft is a large company with a great degree of internal diversity with respect to software engineering practices and domains, and its employees come from a wide array of backgrounds. We intentionally used *Maximum Variation Sampling* [28] when selecting interview participants within this large company. Deploying our survey in additional software organizations and communities will shed light on the generalizability of our findings.

Another risk of this work is that interview participants may have felt pressured to over exaggerate their tasks and how they get them done. In addition, having two interviewers may have intimidated some participants and encouraged them to over emphasize their completed tasks. Including similar questions from our interview on the survey was one approach we took to mitigate potential for biased task explanations in our data.

IX. Related Work

Studying how software engineers approach their work process is the foundation of empirical software engineering [29], [30]. According to Basili, *"The researcher's role is to understand the nature of processes and products, and the relationship between them"* [31]. In software engineering research, the way code is written is a top priority, but so are the software engineers writing it. The works related to this project include how knowledge workers are studied, approach their work, and how software engineers are classified in the practical space.

Characteristics of Software Engineers. Many researchers have demonstrated the different characteristics and working styles of software engineers. Li et al. [32] identified characteristics of developers including personality, decision making skills, ability to engage with a team, and software design approaches to outline what makes a great software engineer. In this work, researchers were able to conduct semi-structured interviews with developers across a large software company to identify 53 attributes of a software engineer. Though their study has breadth, it lacks the details of actions and processes used by software engineers that our work reflects. Researchers have also considered characterizing based on software engineer motivations. Beecham demonstrates that many of these motivators are context dependent on culture, experience, and individual characteristics of software engineers [33]. Though this work acknowledges that context dependent factors may exist, they do not mention how these factors impact types of software engineers work as we show in our work.

Software Engineering Personas. One way to understand how software engineers work is to understand how they fit on their team and the role they play. Zhu et al. [34] identified tools that can help support the development processes based on their role. However, this does not provide fluidity that roles in software engineering are not always clearly defined as Chimalakonda and colleagues found [35]. In the aforementioned work, researchers defined a software engineering ontology that helped define other challenges in understanding software engineering roles. The work by Chimalakonda and colleagues is also motivated by Reinhardt et al. [4] description of knowledge worker roles, but did not identify that the knowledge worker actions that define personas that may exist on a spectrum as we demonstrate.

X. Conclusion

Software engineers are often classified in a single group that do not take their tasks and working style into consideration. To discourage this invalid categorization, we adapted knowledge worker actions to better understand how software engineers at a large software company work. We interviewed and surveyed software engineers across internal organizations to comprehend their tasks, approaches, and who they work with. We used statistical analysis of how software engineers spend their time to provide a data-driven approach to identify clusters and inform our seven personas. We encourage researchers and practitioners to apply these personas to other software organizations to confirm how we can expand these personas and continue to grow informed descriptions of how software engineers work. In concert with prior literature, we are just scratching the surface of how these personas could benefit software engineers and those studying other knowledge workers.

References

[1] J. Zumbrun, "The rise of knowledge workers is accelerating despite the threat of automation," May 2016. [Online]. Available: http://blogs.wsj.com/economics/2016/05/04/the-rise-of-knowledge-workers-is-accelerating-despite-the-threat-of-automation/

[2] T. H. Davenport, *Thinking for a living: how to get better performances and results from knowledge workers*. Harvard Business Press, 2013.

[3] A. Wolkov, "There is only one type of software engineer," Jul 2016. [Online]. Available: https://www.linkedin.com/pulse/only-one-type-software-engineer-adam-wolkov

[4] W. Reinhardt, B. Schmidt, P. Sloep, and H. Drachsler, "Knowledge worker roles and actionsresults of two empirical studies," *Knowledge and Process Management*, vol. 18, no. 3, pp. 150–174, 2011.

[5] A. Cooper *et al.*, *The inmates are running the asylum:[Why high-tech products drive us crazy and how to restore the sanity]*.

[6] M. Rahimi and J. Cleland-Huang, "Personas in the middle: automated support for creating personas as focal points in feature gathering forums," in *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*. ACM, 2014, pp. 479–484.

[7] S. Faily and J. Lyle, "Guidelines for integrating personas into software engineering tools," in *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*. ACM, 2013, pp. 69–74.

[8] J. L. Dupree, R. Devries, D. M. Berry, and E. Lank, "Privacy personas: Clustering users via attitudes and behaviors toward security practices," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 2016, pp. 5228–5239.

[9] E. White, "Who are mort, elvis, and einstein?" May 2006. [Online]. Available: https://blogs.msdn.microsoft.com/ericwhite/2006/05/11/who-are-mort-elvis-and-einstein/

[10] J. Pruitt and J. Grudin, "Personas: practice and theory," in *Proceedings of the 2003 conference on Designing for user experiences*. ACM, 2003, pp. 1–15.

[11] E. Friess, "Personas and decision making in the design process: an ethnographic case study," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1209–1218.

[12] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, "Gendermag: A method for evaluating software's gender inclusiveness," *Interacting with Computers*, p. iwv046, 2016.

[13] N. Marsden and M. Haag, "Evaluation of gendermag personas based on persona attributes and persona gender," in *International Conference on Human-Computer Interaction*. Springer, 2016, pp. 122–127.

[14] T. H. Davenport, "Rethinking knowledge work: A strategic approach," *McKinsey Quarterly*, vol. 1, no. 11, pp. 88–99, 2011.

[15] N. Mundbrod, J. Kolb, and M. Reichert, "Towards a system support of collaborative knowledge work," in *International Conference on Business Process Management*. Springer, 2012, pp. 31–42.

[16] M. Maram, P. Prabhakaran, S. Murthy, and N. Domala, "Sixteen roles performed by software engineers in first one year," in *2009 22nd Conference on Software Engineering Education and Training*, 2009.

[17] D. Ford, T. Zimmermann, C. Bird, and N. Nagappan, "Appendix to personas in practice: Adapting knowledge worker actions to software engineer," https://www.microsoft.com/en-us/research/publication/appendix-personas-practice-adapting-knowledge-worker-actions-software-engineer/, Tech. Rep. MSR-TR-2016-75, 2016.

[18] B. E. Thomas and J. P. Baron, "Evaluating knowledge worker productivity: literature review," DTIC Document, Tech. Rep., 1994.

[19] B. Kitchenham and S. Pfleeger, "Personal opinion surveys," *Guide to Advanced Empirical Software Engineering*, pp. 63–92, 2008.

[20] P. Morrel-Samuels, "Getting the truth into workplace surveys," *Harvard business review*, vol. 80, no. 2, pp. 111–118, 2002.

[21] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[22] K. Goodwin, "Getting from research to personas: harnessing the power of data," https://www.cooper.com/journal/2002/11/getting_from_research_to_perso, 2008.

[23] M. Harper and P. Cole, "Member checking: can benefits be gained similar to group therapy?" *The Qualitative Report*, vol. 17, no. 2, pp. 510–517, 2012.

[24] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *Proceedings of the 2013 international conference on software engineering*. IEEE Press, 2013, pp. 712–721.

[25] T. Barik, R. DeLine, S. Drucker, and D. Fisher, "The bones of the system: A case study of logging and telemetry at microsoft," in *Proceedings of the 38th International Conference on Software Engineering Companion*, ser. ICSE '16. ACM, 2016, pp. 92–101.

[26] P. Sachs, "Transforming work: collaboration, learning, and design," *Communications of the ACM*, vol. 38, no. 9, pp. 36–44, 1995.

[27] R. A. Settersten and G. O. Hagestad, "What's the latest? ii. cultural age deadlines for educational and work transitions," *The Gerontologist*, pp. 602–613, 1996.

[28] M. Q. Patton, *Qualitative evaluation and research methods*. SAGE Publications, inc, 1990.

[29] W. S. Humphrey, *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[30] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, pp. 131–164, 2009.

[31] V. R. Basili, "The role of experimentation in software engineering: past, current, and future," in *Proceedings of the 18th international conference on Software engineering*. IEEE Computer Society, 1996, pp. 442–449.

[32] P. L. Li, A. J. Ko, and J. Zhu, "What makes a great software engineer?" in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015, pp. 700–710.

[33] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *Information and software technology*, vol. 50, no. 9, pp. 860–878, 2008.

[34] H. Zhu, M. Zhou, and P. Seguin, "Supporting software development with roles," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 36, no. 6, pp. 1110–1123, 2006.

[35] S. Chimalakonda and K. V. Nori, "On the nature of roles in software engineering," in *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*. ACM, 2014, pp. 91–94.