Reskilling White-Collar Workers: What's in it for Firms?

by

Frank Ryan

MSci. Computer Science and Philosophy
Durham University, UK (2016)

Submitted to the Institute for Data, Systems and Society in partial fulfilment of the requirements for the degree of

Master of Science in Technology and Policy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author……..
Technology and Policy Program
Institute for Data, Systems, and Society
June 26, 2020

Certified by……..
Elisabeth Reynolds
Executive Director, MIT Work of the Future Task Force
Principal Research Scientist, MIT Industrial Performance Center
Thesis Supervisor

Accepted by…….
Noelle Selin
Director, Technology and Policy Program
Associate Professor, Institute for Data, Systems, and Society &
Department of Earth, Atmospheric and Planetary Sciences

Reskilling White-Collar Workers: What's in it for Firms?

by

Frank Ryan

Submitted to the Institute for Data, Systems, and Society on June 20, 2020 in partial fulfillment of the requirements for the degree of Master of Science in Technology and Policy

Abstract

Increasing rates of skill obsolescence amongst white-collar workers resulting from rapid technological and organizational shifts within firms, coupled with an insufficient supply of workers proficient in technical areas such as software engineering and data science, has meant that firms are beginning to look to reskilling their incumbent workers as a means of filling skill gaps. This aim of this study was to examine the process and outcomes of efforts made by a US-based diversified global insurance company between 2016 and 2018 to reskill and reassign approximately 300 employees in the face of changing technological and market demands. More specifically, this involved running an immersive coding program for employees with little or no prior coding experience, with the ultimate aim of finding them new job roles in technical areas such as software engineering and data science. Quantitative analysis of longitudinal employee data, combined with qualitative interviews with program participants and administrators of the program, was used to assess the wide-ranging impacts that program had both on the firm and its workforce. Results suggest that the firm benefitted immensely from the retention of business-related knowledge and a low turnover rate of program graduates compared to externally hired software engineers. The majority of program participants also benefited from receiving free technical skills training, resulting in improved career prospects, however inefficiencies in the onboarding process for placing code school graduates on to new technical teams meant that some unfortunate participants were unfairly disadvantaged. Overall, this study provides encouraging evidence for the potential for white-collar reskilling programs to bring about positive outcomes for both firms and their workers, with several lessons learned for future implementations of similar programs.

Table of Contents

Chapter 1: Introduction

Thousands of companies across the United States are in the thick of a digital revolution that is requiring them to rapidly transform their operations. These organizational developments, combined with rapid technological advancements in areas like machine learning and cloud computing that facilitate the automation of tasks, is resulting in skill obsolescence within the workforce at an ever-increasing rate (De Grip and Van Loo, 2002). As many as 375 million workers — or roughly 14 percent of the global workforce — may need to switch occupational categories by 2030 as digitization, automation, and advances in artificial intelligence disrupt the world of work (Manyika *et al.*, 2017). The kinds of skills companies require will shift, with profound implications for the career paths individuals will need to pursue.

When faced with the threat of skill obsolescence within its workforce, one course of action that a firm can take is the traditional "Fire and Hire" approach, whereby employees with redundant skills are laid off and new employees with newly desired skillsets are hired by the firm. This approach has become the stock solution for skill obsolescence for firms, as shareholders can more readily understand a plan that calls for simultaneous layoffs and hiring rather than a resource-intensive reskilling initiative (Bidwell *et al*., 2013). In reality however, hiring fresh talent in response to skill obsolescence can also be expensive, and layoffs can incur millions of dollars in severance and restructuring costs (Bidwell, 2011). This is especially true in tight labor markets where the supply of skills like cloud computing, cybersecurity and data science is far outstripped by the immense demand for them. Furthermore, terminating workers with outdated skills also results in negative outcomes for the individuals who lose their livelihoods as a consequence.

Alternatively, a firm can choose to go down the reskilling route, which involves investing time and money in to training incumbent employees for a new role within the company through teaching them newly desired skills. This is likely to result in markedly better outcomes for the workers involved in the reskilling process as they are able to keep their jobs and have been equipped with a new set of marketable skills, usually free of charge. However, it is also clear that firms incur a number of risks in opting for this policy as it is the employer that generally bears the greater financial burden as they pay most of the direct monetary costs of reskilling. For instance, it may transpire that some employees are unable to be trained to an acceptable standard, perhaps due to a lack of natural ability or desire to learn them, in which case the firm will not see a return on its investment. The firm also bears the cost of the forgone loss in productivity as a result of employee time spent on the reskilling program. Alternatively, the newly reskilled worker may decide to leave the firm after being reskilled as they are now able to attract hire paid work elsewhere (Mohrenweiser et al., 2019).

What possible reasons might there be for a firm to invest time and money in to reskilling its workers, as opposed to hiring new ones? This, as its title may suggest, is one of the key questions which this thesis seeks to address. Furthermore, if a firm does decide to go down the reskilling route: how can they go about doing it effectively? What effects - positive or negative - can reskilling programs have on the firm, the worker and, most importantly, society as a whole? Finally,

how do firms, workers and society more broadly adapt to the changes that loom important going forward and what can be done to improve outcomes for both firms and workers? It is hoped that, in providing satisfactory answers to these questions, the conclusions drawn from this thesis can serve as a guide for firms wishing to undergo the reskilling process.

Outline of Case Study

As a means of finding answers to the questions posed above, this thesis will closely examine the process and outcomes of efforts made by a US-based diversified global insurance company between 2016 and 2018 to retrain and reassign approximately 300 employees in the face of changing technological and market demands. More specifically, this involved running an immersive coding program for employees with little or no prior coding experience, with the ultimate aim of finding them new job roles in technical areas such as software engineering and data science.

One of the fundamental objectives of the program was to find new roles within the firm for employees whose jobs were particularly under threat due to skill obsolescence resulting from a recent transition to Agile software development. The Agile approach to development emphasizes the incremental delivery of a software product whilst it is still being built, allowing for continuous feedback from the end-users during the development process (Beck *et al*., 2001*)*. As will be discussed in further detail in Chapter 3, Agile development teams are typically smaller and highly cooperative compared to that of other software development models in order to facilitate quick decision-making and multiple design iterations.

The firm defines a reduction in force (RIF) as the elimination of a position with no intention of replacing it, resulting in a permanent cut in headcount. If a given employee is deemed to be "RIF-eligible" – a term that will be used frequently throughout this paper - it means that, whilst they have not officially received a termination notice, their position at the company is under threat and there is a reasonable probability of that employee being terminated in the near future.

Methodology

Data collection consisted of both quantitative and qualitative methods. Quantitative analysis used longitudinal personnel data to understand whether certain employee characteristics were in any way correlated with the outcomes of the reskilling process. Further details on the type and structure of the data that were used are provided in Chapter 5. Qualitative work involved semi-structured interviews with 21 of the firm's employees that were either staff members involved in the design and execution of the reskilling efforts or employees that participated and successfully graduated from the program. Specifically, this included:

- The firm's Chief Information Officer
- A senior HR officer who was responsible for planning and overseeing the implementation of the program
- A Program manager that was responsible for mentoring participants whilst they were attending the code school and placing them in new teams upon graduation

- 3 employees in senior technical roles (e.g. IT Manager or Principal Software Engineer) that supervised several program participants after they graduated from the program
- 4 interviewers for the program
- 11 program participants

Apart from the CIO and the senior HR manager, all interviewees were selected by the firm.

This Paper

The remainder of this paper is structured as follows: Chapter 2 provides a brief literature review of workplace training programs, where I identify a gap in the literature relating to employer-provided technical reskilling; Chapter 3 provides detail on the organizational developments that led to the firm implementing the technical reskilling program; Chapter 4 describes each of the program execution phases, beginning with the recruitment and interview process and ending with a summary of the onboarding period; quantitative and qualitative analysis of the outcomes of the program are discussed in Chapters 5 and 6 respectively; finally, Chapter 7 provides some concluding remarks along with discussion of potential avenues for future research.

Chapter 2: Literature Review

There is a considerable amount of academic literature assessing the impact of employer-provided "training", with noticeably less attention paid to employer-provided "reskilling". Whilst reskilling can certainly be classified as a type of training, the majority of instances of employer-provided training examined refers to what is most commonly referred to as "upskilling": the development of a worker's skills within their current line of work in order to enable them to do their job more effectively. Contrastingly, in the case of reskilling, the objective is to equip the worker with new skills in order for them to do a different job. For example, providing a Software Engineer with training to learn a new programming language can be considered "upskilling", as the employee already knows how to code and therefore the training is designed simply to enhance their skillset in relation to their current job role. In contrast, training a sales representative with minimal prior technical knowledge how to code using programming languages in order for them to become a software engineer can be classified as reskilling.

Human capital theory predicts that employer-provided training, in its most general sense, can have a positive impact on wages as long as it increases productivity (Haelermans and Borghans, 2012). However, wage increases due to training may be lower than the accompanying increases in productivity when the wage structure of the firm is more compressed and inflexible (Acemoglu, 1999). Overall, research relating to organizational-level benefits from employer-provided training is not nearly as abundant as the literature on individual employee benefits.

Not only have there been relatively few empirical studies demonstrating organizational-level impact, but those studies that have been done typically use self-reported data and provide a tenuous causal link back to the training activities (Tharenou et al. 2007). Using a field experiment with random assignment to a training program for call center workers, Sauermann (2015) finds that workers reciprocate employer training investments by making greater effort and increasing individual productivity. Similarly, this positive reciprocity can lead to employees putting off retirement as a response to training opportunities when it is seen as a credible gesture to facilitate employees' future employability (De Grip, 2019). In terms of the social benefits of training, the general consensus among economists is that efforts to train workers lead to improvements in the quality of the labor force, which in turn is one of the most significant contributors to economic growth (Becker, 1962).

When it comes to reskilling, there have been very few studies that provide thorough analysis into programs aimed at equipping workers with new employable skills in response to skill obsolescence. Even amongst these few instances of reskilling, there are no cases of employer-sponsored reskilling programs, as all prior studies involve implementation by government or public-sector actors. Jacobson *et al.* (2005) provide a comprehensive examination of the impact of community college-based reskilling on displaced workers earnings in Washington State. They find that participation in the reskilling program raised individuals' earnings by approximately 12% for men and 15% for women. This impact reflects the gains from a single academic year of schooling in technical vocational courses, academic math and science classes, and health-care programs.

These results are consistent with the human capital literature that indicates that one year of formal schooling raises a student's subsequent earnings by approximately 10% per year (Heckman et al., 2003).

Another study examines the reskilling program that resulted from the closure of the Anglo-Australian steel manufacturing firm BHP Steel, leading to the retrenchment of roughly 1,800 BHP employees and another 1,000 contractors (Callan and Bowman, 2015). The displaced workers were provided funding from the New South Wales Government to enroll in training courses to allow them to reskill in areas such as computer programming and hospitality services for up to 12 months. In total approximately 7,000 separate training events were funded through the program and roughly 90% of participants found new employment. As a result, this reskilling intervention, officially known as the Pathways Program, has been recognized as setting an international standard and recommended as the benchmark for outplacement programs.

Beyond the academic literature, there have been several recent high-profile cases of reskilling endeavors launched by firms. Amazon.com will soon begin offering a path for workers to rise from warehouses to data centers, doubling their earnings along the way and helping Amazon staff a fast-growing part of the business (Amazon Inc, 2020). Elsewhere, Disney's CODE: Rosie program gives women employees in non-technical roles an opportunity to switch careers. The program begins with three months of training in basic computer science concepts and lessons in how to code using programming languages such as Python. The program participants - dubbed "Rosies" - then segue into a yearlong apprenticeship consisting of two six-month chunks in different teams within the company before eventually taking a job within one of Disney's technical groups (McCracken, 2019).

Despite the abundance of prior studies examining the various impacts of workforce training, or "upskilling", very little attention has so far been given to the potential for reskilling as a means for filling skill gaps and reducing skill obsolescence amongst workers. Furthermore, the handful of studies that are devoted to reskilling tend focus on the reskilling of workers from typically blue-collar industries such as manufacturing. Recent anecdotal examples of technical reskilling for white-collar workers, such as Disney's CODE:Rosie program are yet to be validated by the scientific literature. Thus, as far as the authors are aware, this study is the first to conduct an in-depth analysis of a reskilling program aimed at providing high-level technical coding skills to white collar workers.

Chapter 3: Motivation Behind the Reskilling Program

Transition to the Agile software development model

The main impetus for the firm's decision to reskill its workforce was a perceived need to completely overhaul its software development process. Up until 2016, the company produced internal and external software under a model known as the "Waterfall" model whereby a given project - such as the development of a new application to be used by underwriters - is broken down into linear sequential phases, with each phase corresponding to the specialization of a particular task. Phases are carried out one at a time and in order, with each phase depending upon the deliverables of the previous one. Figure 1 illustrates a series of development phases present in the typical Waterfall model. Otherwise known as the Linear Sequential model, it was the first process model to be applied to software development (Royce, 1987), having originated from the manufacturing and construction industries, where highly structured physical environments mean that design changes becoming prohibitively expensive early on in the development process.

One of the advantages that the model provides is that it allows for the specialization and isolation of tasks, whereby each phase is typically executed by separate teams within the firm. For example, to walk through the process outline in Figure 1, during the initial Requirements phase a team of Business Systems Analysts utilize their business-specific expertise and consult with the users of the software to decide upon and document the requirements for a project. These are then passed on to the system architects who map out the high- and low-level designs during the Design phase. Development involves software engineers taking the design documents and building it, through the use of programming languages, so that it follows the design specified by the system architects. The testing team then tests the complete application and identifies any defects that may need to be fixed by the developers before it is deployed. The operations stage involves not just the deployment of the application, but also subsequent support and maintenance that may be required to keep it functional and up to date.

Under this model, each team member is specifically trained to carry out the role required for their specific phase of development. The sequential nature of the model means that an employee or group of employees working on a particular phase of the process is isolated from what is going on in all other phases, allowing for employees to specialize and become experts within their given role. For instance, as software engineers receive detailed specifications that have been passed down from the requirements and design phases, they are not required to have an in-depth knowledge of the business or industry context of the software that they are developing and are therefore able to quickly switch between projects. Similarly, the role of the Business Systems Analyst is to produce a requirements document that defines *what* the application should do, but not specifically *how* the software should be designed and developed. Whilst it is necessary for them to have a solid understanding of what is possible from a software development standpoint and be able to understand and technical jargon in order to effectively with technical experts, they are not expected to have an in-depth knowledge of architectural styles and design patterns or be able to code using programming languages.

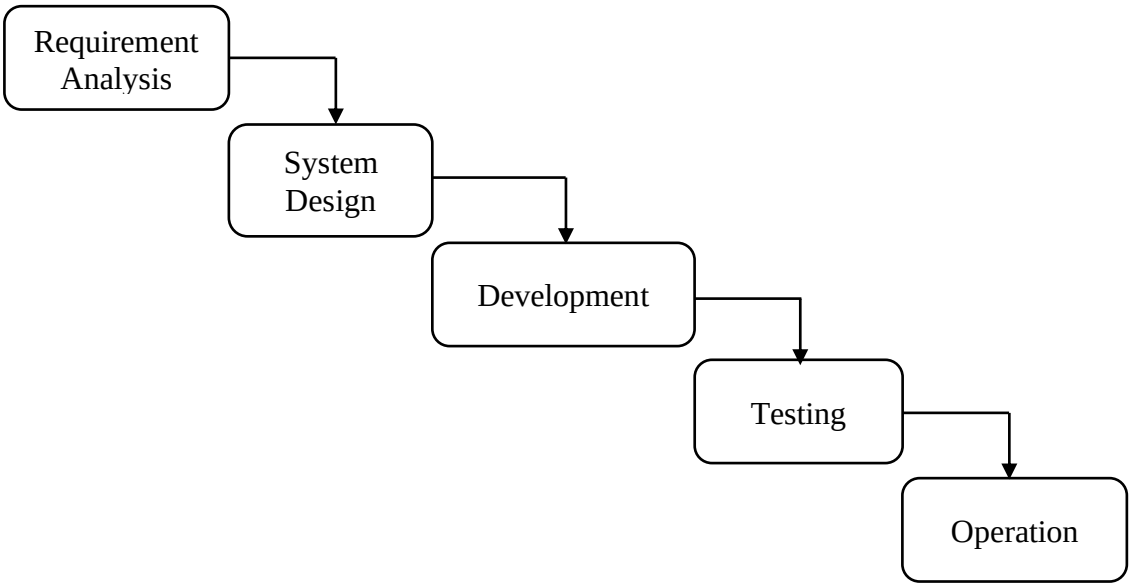

Figure 1: A typical Waterfall Model

This traditional approach to software development does come with a number of drawbacks, however. Perhaps the most damning feature of the model is its inherent lack of adaptability across each of the development stages. If a fundamental flaw in the design of the system were to be revealed during the testing phase, it would not only require a dramatic leap backward in stages of the process, but in some cases, can often lead to a devastating realization regarding the legitimacy of the entire system. Delaying the testing phase until after the entire application has already been built not only means that most bugs or even design issues will not be discovered until very late into the process, but it also encourages lackadaisical coding practices since testing is only an afterthought. Similarly, due to the strict step-by-step process that the waterfall model enforces, another particularly difficult issue to get around is that user or client feedback which is provided at the very end of the development cycle can often be too little, too late. While project managers can obviously enforce a process to step back to a previous stage in response to an unforeseen requirement or change coming from a client, it will be both costly and time-consuming, for both the development team and the client.

Over time, the insurance firm began to realize that this traditional, stratified, Waterfall model of development was far from optimal. Due to the size of the company, and the number of stakeholders involved in developing new software applications, it would take months to specify requirements and come up with a design for a project and then a year or more to develop and test it. By this point, it was not uncommon for the user requirements to have changed or the technology to have gone out of date, thus rendering the product not up to standard. In an interview, the firm's Chief Information Office described this issue as follows:

> *"We had to move away from multi-year, billion-dollar programs that promised nirvana three years hence. We had to move toward something with shorter lag times. Technology shifts would take too long; by the time the projects changed it would be no longer really needed."*

As a result of evident shortcomings of the Waterfall model, in 2016 the firm made the decision to gradually phase out its usage for a more modern software development approach that was better suited to their needs: the Agile development model. The *Manifesto for Agile Software Development* (Beck *et al*., 2001) was written by a group of seventeen software leaders – the self-titled "Agile Alliance" – who had become frustrated by the long lead times and documentation overheads involved with Waterfall. Agile development has since become the most widespread approach to software development, with an estimated 86% of professional software developers using agile in their work in 2018 (Kukhnavets, 2019). Instead of in-depth planning phase at the beginning of a project, Agile methodologies were designed with the primary aim of being open to changing requirements over time and to encourage constant feedback from the end users. Rather than following a series of rigidly pre-defined steps, development is carried out in iterative cycles of designing, building, and testing, with the goal of each iteration to produce a Minimum Viable Product: a version of the desired product, with just enough features to satisfy early customers and provide feedback for future product development (Moogk, 2012). Each iteration is intended to be small and easily manageable to be completed within a couple of weeks only. Figure 2 presents a typical iteration cycle.

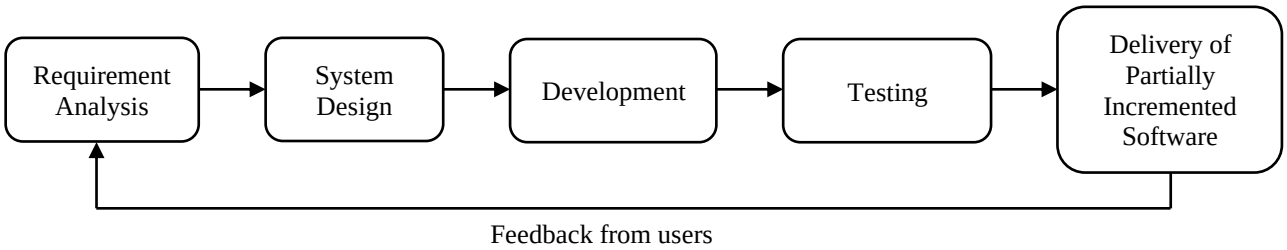

Figure 2: A typical Agile Development Cycle

Today, roughly 90 percent of teams within the firm operate in an agile fashion. Rather than issuing a directive that a project must be complete by the end of the year, they now align a number of agile teams associated with driving key business metrics, such as customer satisfaction or penetration of digital technologies for claims. Every few weeks, the teams evaluate if they are building the right technologies and receive feedback from customers directly with regards to what is working and what is not. Consequently, the transition from Waterfall to Agile development has led to a significant improvement in the speed at which the firm is able to deploy, scale, and fix its software applications.

The transition to Agile development has also resulted in equally seismic shifts in the composition of software development teams and the blend of different skillsets within them. Whilst Waterfall allowed for specialized teams to work on separate phases of the development

process, Agile has introduced a need for smaller, multi-disciplinary employee groups that collaborate on new products more efficiently, bridging the gap between the IT and business arms of the firm. As agile teams are typically smaller and highly cooperative in order to facilitate quick decision-making and multiple design iterations, all team members, including software engineers, therefore need to be well-versed in both the business context and technical constraints. Strong people skills, such as effective verbal communication and the ability to work within a team, are also a must for all team members so that they are able to fluidly cooperate with one another. Furthermore, the need for Agile teams to be co-located, in order to facilitate seamless communication between members, has meant that all employees involved in the software development process are now required to live within a commutable distance of one of the firm's four "technology hubs" within the United States.

The shift in skill requirements that accompanied the agile transformation has led to the firm eliminating more business-centric roles that are involved in the software development process, such as Business System Analysts in favor of developers who can fix system errors as customers report them. In the time of the waterfall model, business systems analysts were tasked with consulting with managers or other leaders of functional areas to understand how they use IT systems and then translating those needs into a concrete plan for the software development teams to implement. However, under the new customer-centric agile system, software architects and engineers take on some of the responsibilities of business system analysts by learning to talk to the users without a middleman. This means that the firm now has a greater need for technical experts such as software engineers and software architects, whilst allowing for the elimination of analyst roles that do not fit in to the agile model.

Analyst roles were by no means the only job family that were set to be axed between 2017 and 2020. Project Managers, who under the Waterfall model were in charge of overseeing the execution of the Waterfall process from start to finish, were replaced by "scrum masters" and "product owners", who take on many of these managerial roles within an agile team. The role of the scrum master is to motivate and lead the team whilst maintaining efficiency for timely completion of a given project. Product owners look at the project from the customer's perspective and are responsible for managing product backlogs (the priority list of user requirements).

Figure 3 visualizes the effect that the agile transition had on various different job titles at the firm. Here it can be clearly seen that analyst roles suffered significant reductions in headcount between 1/31/2017 and 4/30/2020. Most impacted was the business systems analyst role, which fell from 1205 employees to just 25 during this period. Traditional project management positions also suffered visible losses whilst product owners and scrum masters went from 0 employees in January 2017 to 532 and 303 respectively in April 2020.

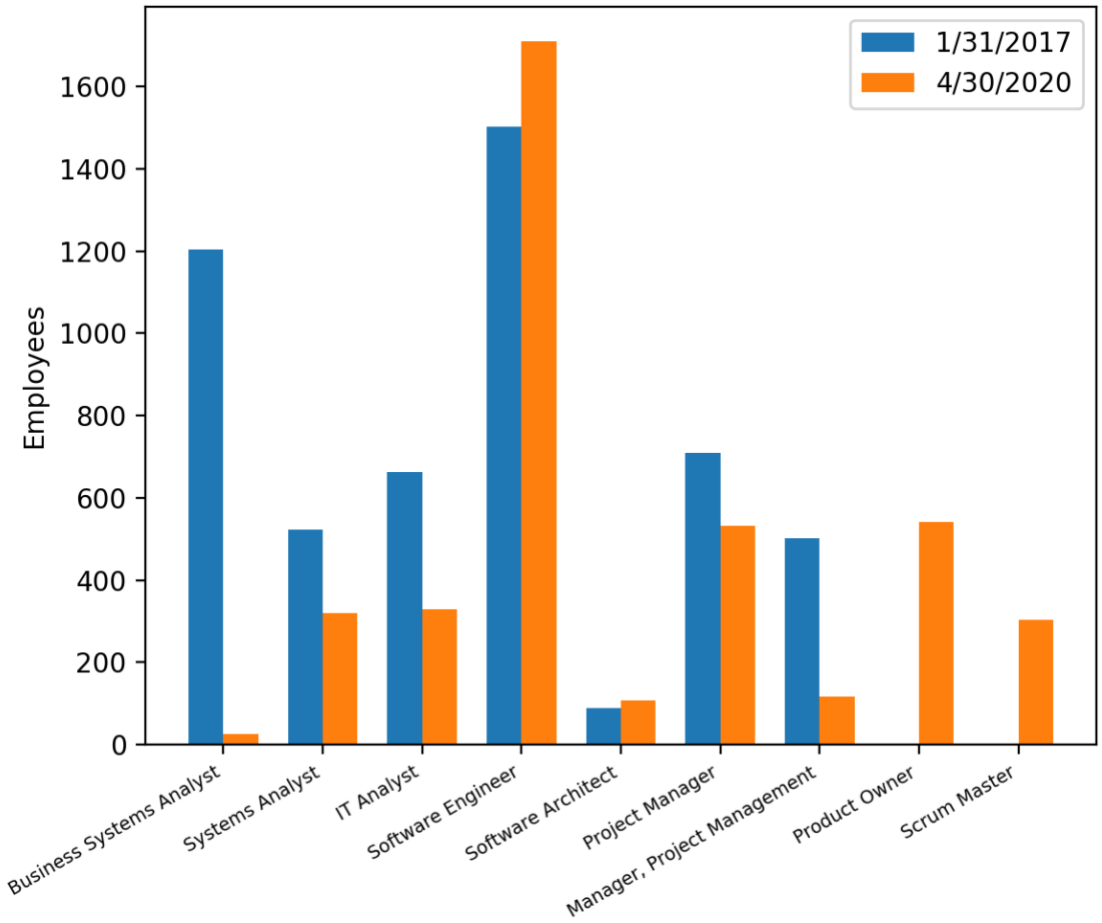

Figure 3: Employee headcount by job title and date

Desired Skill Areas

Whilst numerous employees faced skill obsolescence as a result of the transition into Agile development, the digital transformation that accompanied the transition created a demand for technically focused skill areas within the firm. These include:

Back-end development – The back end of a website consists of a server, an application,and a database, all three of which are never directly seen by the user. A back-end developer builds and maintains the code that powers these components which, together, enable the 'behind-the-scenes' functionality of web applications. In order to make the server, application, and database communicate with each other, back-end developers use server-side languages such as PHP, Ruby, Python, and Java to build an application, in addition to tools like MySQL, Oracle, and SQL Server to find, save, or change data and send it back to the user in front-end code. In our case study, the majority of the web applications built by the company are developed using the Java programming language. As one the world's most popular programming languages (TIOBE, 2020), Java has been in use for almost 30 years, and is particularly popular with desktop and business software developers.

Front-end development - The front end of a website is the part that users interact with, Front-end developers are responsible for writing the code that determines how the site looks to users. In order to do this, front-end developers must be proficient in three main languages: HTML, CSS, and JavaScript. Effective front-end developers also need to be familiar with frameworks like Bootstrap, React and AngularJS, which ensure aesthetically pleasing content irrespective of the device being used, and libraries like jQuery and LESS, which package code into a more useful, time-saving form. Using these tools, front-end developers work closely with designers or user experience analysts to bring mockups, or wireframes, from development to delivery.

User Interface/Experience Design (UI/UX) – A user interface is the point of interaction between the user and a digital device or product. In relation to websites and apps, UI design considers the look, feel, and interactivity of the product, by thinking about the optimal use of icons, buttons, color schemes, spacing, and imagery. In contrast, a UX designer thinks about how a user's experience with a product makes them feel, and how easy it is for the user to accomplish their desired tasks. For example: Does the firm's online app make it easy for users to submit insurance claims and manage their pending claims? Whilst technically being two separate fields, UI and UX are often conflated and many companies will deliberately seek out versatile designers who can cover both UX and UI. This is particularly true in the case of our case study, where UI and UX are relatively new areas of employment only recently moved in-house after having previously been outsourced to external contractors.

Data Science – Regarded by many as the "fourth paradigm" of science (Hey, 2009), data science aims to make sense of raw data through uncovering hidden patterns that can be used to glean insights into complex systems. Due to being applicable to any domain that involves vast quantities of data – whether structured or unstructured – it is now being used by virtually all major organizations in some form. Professional data scientists are required to having strong coding abilities in at least one statistical programing language, such as R or Python, and database querying languages such as SQL. A solid understanding of statistics, multivariable calculus and linear algebra, all of which form the basis of many data analysis techniques, is generally also required.

The gradual shift from Waterfall to Agile software development that occurred from 2017-2020 thus led to a significant shift in skill requirements within the technology arm of the company. This resulted in a decline some positions – predominantly analyst roles that had previously provided business-specific expertise under the Waterfall model – and the creation of Agile-specific job roles such as Product Owner and Scrum Master. The underlying aim of the program was therefore to respond to this shift in skill requirements by reskilling workers in these at-risk positions, equipping them with skills relating to front-end and back-end software development, UI/UX design or data science to facilitate their transition in Agile development teams.

Chapter 4: Implementation of the Reskilling Program

The reskilling program ran for a total of 12 months for each participant, with the first three months spent at the code school followed by nine-month onboarding period. During this time, code school graduates were placed on to provisional teams within the company and had their performance assessed. Whilst taking part in the program, participants were paid the same salary as they had been in their previous role at the firm. This chapter will detail each of the program execution phases, beginning with the recruitment and interview process and ending with a summary of the onboarding period.

Pilot Program

The first four cohorts of the reskilling program were implemented as a pilot in order to test the feasibility of running the program before it was implemented at scale. These initial cohorts ran from late 2016 to early 2017 and involved a limited number of participants (3-5 per cohort). Unlike subsequent cohorts, these early iterations of the program were not widely publicized throughout the company and participants were pre-selected. Employees needed to be nominated by their manager in order to apply for the program and were selected based on their job title, job grade, and recent performance ratings.

Recruitment and Interview Process

Following the initial pilot cohorts, application to the program was then opened up to the company as a whole. The program was advertised via company-wide emails and on the front page of the employee web portal. All company employees were eligible to apply for the program, regardless of their job title, length of tenure at the company and whether they were RIF-eligible or not. In total, 1,513 employees applied to the program between 2016 and 2018.

All applicants were required to submit their resume along with a cover letter through an online application portal. Following this, most applicants would be scheduled for an informal telephone call lasting approximately 5 minutes with a member of the HR/recruitment team to explain and clarify what the program would involve if they were selected. For instance, applicants were asked whether they would be comfortable with relocating to a different part of the country whilst attending the code school; and were typically informed that, by taking part in the program, they would effectively be resigning from their current role in the firm. Thus, moving forward in the program involved considerable risk on the part of the employee.

Following this, applicants were required to attend 1-2 in-person interviews which were conducted by senior employees within the company, rather than HR professionals. Each interview usually involved two interviewers, with one interviewer coming from a technical role (such as a principal software engineer) and the other with a managerial background (such as a project manager). This was done with the aim of assessing both a candidate's technical capabilities as well as their interpersonal skills, i.e. the ability to communicate effectively or work within a team. At the end of each interview, the interviewers would compare notes and fill out an interview

assessment form to be sent back to the HR/recruitment team, who would make the final determination as to whether or not to hire the candidate.

Selection Criteria

In terms of technical capabilities, applicants were not expected to have any prior knowledge or experience relating to the technical area (software development, data science etc.) that they were applying for, although this was certainly considered a helpful bonus. Instead, interviewers were instructed to evaluate how the applicant was utilizing their current technical skillset and how they had demonstrated technical competence in the past, through the use of Microsoft Excel for instance. The ability of a candidate to identify and solve problems and adapt to new and unfamiliar situations, such as having to quickly learn a new skill, were also an integral part of their technical assessment.

A candidate's motivation for applying to the program was also thoroughly examined during the in-person interviews. Candidates that were able to articulate why they had chosen the technical area that they had applied for and how they thought that they would benefit from the program in the future were looked upon favorably. Applicants that showed a strong desire to reskill and learn more about their chosen technical area were preferred over others that claimed to be applying for the program more or less just to save their job and remain at the firm. The following quotes from interviews with employees that were responsible for screening and interviewing candidates for the program exemplify the selection criteria used for the program:

> *"If I were interviewing a non-tech person, if they couldn't articulate problem solving then I would use that as an indication of perhaps not being able to excel as a developer. If you can't explain to me what your role is that's a communication weakness."*

> *"In general, I gave the thumbs down to people who didn't appear to have good study habits or learning capabilities, such as the ability to problem solve. I would try and hone in on their ability to independently solve problems and be self-sufficient, as being extremely independent is a very valuable skill for a software developer to have. Especially in a program like that where everyone has different backgrounds and learning styles, you almost need to adapt the curriculum to yourself. If you fell behind it was hard to catch back up. People that I felt were not well suited were those who didn't seem very proactive in their learning."*

> *If the answer to the question "what interests you about this role?" would be "well I have to do something or else I'll lose my job", that isn't really the face that you want to put on when you're applying for a highly sought-after program. They are not showing the initiative to be part of that team and be successful.*

> *"We were shifting away from the traditional waterfall development model to agile practices, so we wanted people to come in and not just be a task worker. We wanted people to be able to sit next to each other, point at something over someone's shoulder and say, "hey have you thought about*

> *this?" or "what does that piece of code do?". It's really about getting people to work together and think: how do we develop a better product?"*

Code School

The firm contracted one of four independent code school providers (Software Guild, Iron Yard, General Assembly and Galvanize) to organize and run each of the 18 code schools in a variety of different locations throughout the country. Each code school would typically run for a total of 12 weeks, and the majority of code schools assigned 1-2 weeks of introductory reading and assignments which participants were required to complete before the commencement of the program. As the firm has numerous offices spread out across the country, the code schools were equally well dispersed, occurring in seven different US cities. Participants that needed to travel in order to attend the code school were provided free return flights from their hometown to the city where the code school was located, in addition to free accommodation and a living allowance for the entire 12 weeks.

Whilst the course content provided within each cohort varied based on the cohort type and code school provider, the workload and amount of contact hours appeared to be relatively consistent across cohorts. Students were required to attend the code school from 9am until 5pm every weekday and homework was assigned for evenings and weekends. 9 out of the 11 participants interviewed described their code school experience as "difficult" or "challenging", whilst over half reported experiencing some degree of "imposter syndrome". Despite the demanding nature of the code school, 10 of the 11 interviewed participants still reported either enjoying the program or having a positive experience overall.

Due to the time commitment that the code school entailed, participants that were required to travel to a new city to attend the code school spoke of how being "isolated" away from their daily lives and usual responsibilities, such as looking after their children, helped to limit distractions and allowed them to immerse themselves within the course, thus getting more out of it than they would have done if they had attended the code school in their city of residence. Participants that were required to travel in order to attend the program also credited a high level of camaraderie with other team members of the code school to their success on the program.

Table 1 provides a breakdown of each of the 18 cohorts run as part of the reskilling program; it includes the cohort start date, cohort type (front-end, data science, etc.), the number of participants on that cohort, and several measures of participant satisfaction. Participant satisfaction survey response options were provided on a five point scale, with the options being: "Very satisfied", "Satisfied", "Neutral", "Dissatisfied", "Very Dissatisfied" (for the "More Difficult that Expected" survey question, the options were instead: "Strongly agree", "Agree", "Neither agree nor disagree", "Disagree", and "Strongly disagree"). Participants that responded as either "satisfied" or "very satisfied" were categorized as satisfied, thus allowing for a percentage of satisfied/dissatisfied participants to be calculated and presented in the table. Cohorts marked with "N/A" for each of the satisfaction scores did not have satisfaction surveys conducted.

Onboarding Process

Upon completion of the program, participants were due to be assigned technical roles within teams across different areas of the company. Applicants were made aware in advance of applying that they would be required to live within a commutable distance of one of the firm's four "technology hubs" within the United States. The first nine months after the code school were considered an "assessment period", after which point the participant would be evaluated by their manager and other technical managers to determine their new job grade, job title and salary. In some cases, the participant may even transfer to a new team, however this generally did not happen if they were deemed to be a good fit for their originally assigned team. During the code school and nine months following, the participant's salary was held at the same level as it was prior to them undertaking the program. However, participants were not guaranteed to be given the same salary or job grade upon completion of the program.

The length of time taken to assign program participants on to new teams after the code school varied considerably. Most applicants were notified of which team they would be joining at some point during the three months at which they were at the code school, or soon afterwards. However, the firm had some difficulty in finding a sufficient number of vacancies on teams to match the number of applicants coming out of the code school which meant that several participants were left without teams, some for up to an entire year. Those not placed on teams were either switched between teams to temporarily work on projects or given individual assignments to work on.

This chapter has described the journey that a given employee would have taken through the 12-month process of completing the reskilling program; from the recruitment and interview phase, to the code school, the onboarding process, and finally the 9-month assessment period. The detailed overview of how the program was run, as provided by this chapter, will allow the reader to understand the analysis of the program outcomes presented in subsequent chapters.

Start Date	*Type*	*Code School*	*Number of Participants*	*Prework Satisfaction*	*Instructor Satisfaction*	*Expectation Satisfaction*	*More Difficult Than Expected*	*Onboarding Satisfaction*
9/6/2016	Data Science	Galvanize	5	N/A	N/A	N/A	N/A	N/A
9/12/2016	Back-end	Software Guild	3	N/A	N/A	N/A	N/A	N/A
9/26/2016	Front-End	Iron Yard	3	N/A	N/A	N/A	N/A	N/A
10/10/2016	Front-End	General Assembly	3	N/A	N/A	N/A	N/A	N/A
2/6/2017	Back-end	Software Guild	21	75%	75%	76%	62%	57%
3/20/2017	Front-End	General Assembly	29	66%	69%	76%	55%	44%
4/17/2017	Back-end	Software Guild	19	26%	74%	58%	68%	42%
5/15/2017	Front-End	Iron Yard	14	N/A	N/A	N/A	N/A	N/A
5/15/2017	Back-End	Iron Yard	11	N/A	N/A	N/A	N/A	N/A
6/26/2017	UI/UX	General Assembly	20	47%	94%	95%	21%	32%
7/10/2017	Back-end	Software Guild	20	37%	89%	58%	63%	37%
8/7/2017	Front-End	Iron Yard	18	53%	53%	53%	7%	34%
8/7/2017	Back-End	Iron Yard	21	43%	100%	93%	14%	28%
9/11/2017	Front-End	General Assembly	29	64%	84%	68%	28%	20%
10/2/2017	Back-end	Software Guild	21	47%	84%	79%	68%	16%
11/6/2017	Back-End	Iron Yard	17	N/A	N/A	N/A	N/A	N/A
1/22/2018	Back-end	Software Guild	18	58%	94%	100%	29%	71%
3/19/2018	Back-end	Software Guild	16	N/A	N/A	N/A	N/A	N/A
Mean:			**16**	**52%**	**82%**	**76%**	**42%**	**38%**

Table 1: Summary of Program Cohorts

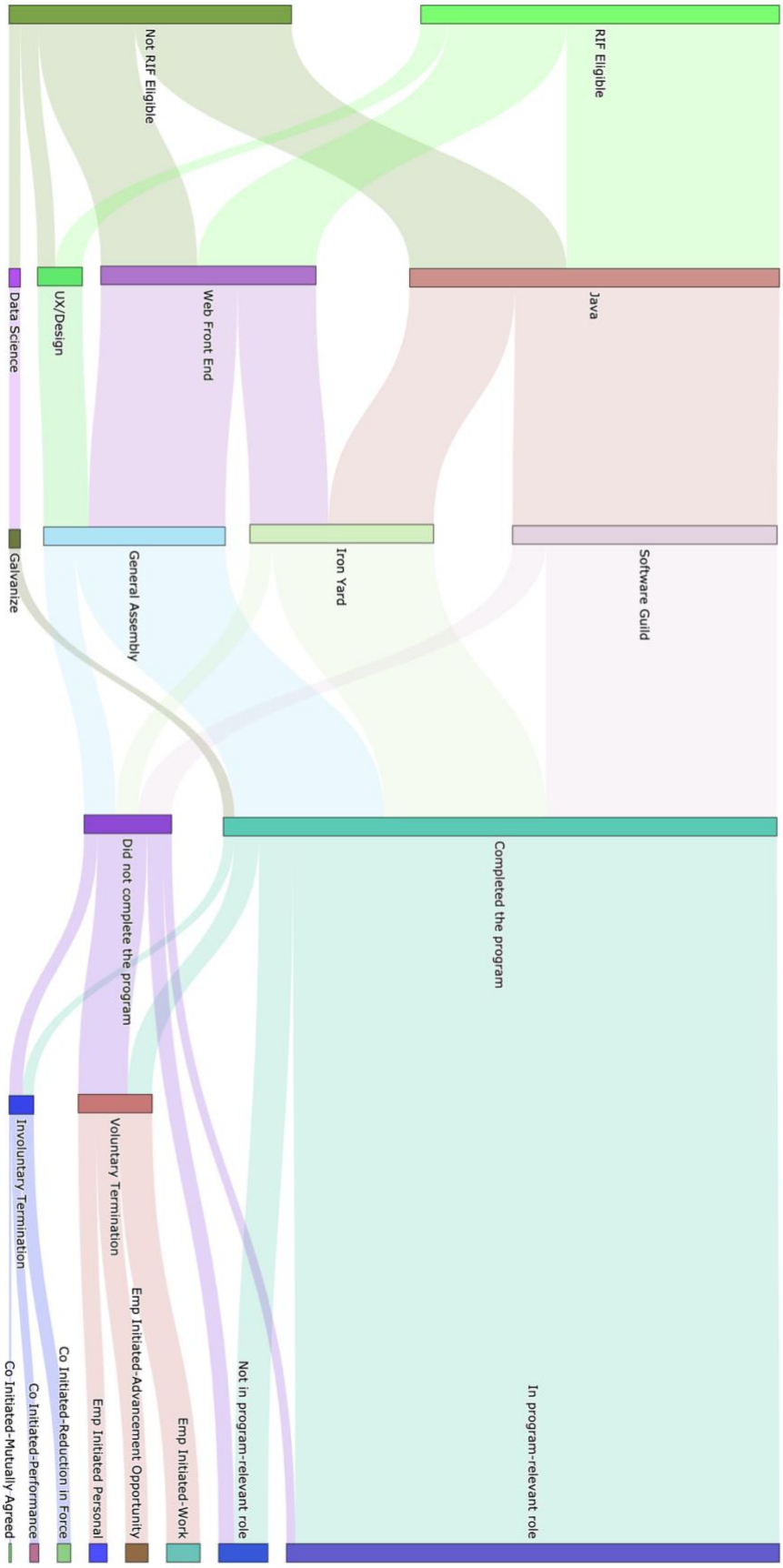

Figure 4: Sankey Diagram of outcomes for all program participants

Chapter 5: Quantitative Analysis of Participant Performance and Outcomes

The firm provided a rich dataset of employee data for all applicants to the program in addition to all employees within the firm that were deemed to be "RIF eligible", amounting to a total of 3,148 rows, with one row for each employee. As one might expect, there was a large amount of overlap between these two groups of people, with many RIF-eligible choosing to apply to the program. The dataset is accurate as of 12/1/2019 and contains a range of variables for each employee, ranging from demographic data – such as age, gender and minority status – to the employee's education level, job title and job grade before and after the program.

The majority of the analysis that is presented in this chapter was carried out by the authors of this paper using the dataset that is described above. However, there are a small number of figures presented here that were created and provided by the firm as they are used summarize sensitive or private data that was not included within the dataset given to us. For instance, Figure 10, Figure 12 and Figure 13 compare the attributes of program participants with that of externally hired software engineers. Data for externally hired software engineers was not included within our dataset so this analysis was conducted by the firm.

A high-level view of the recruitment and outcome funnel for the program is presented in Figure 5. Across the 18 cohorts that were run between 2016 and 2018, the program received applications from a total of 1,513 employees, some of whom applied on multiple occasions. Of these, 307 applicants received offers, which were accepted by 289, culminating in a total of 248 employees successfully completing the 12-month program.

The Sankey diagram shown in Figure 4 provides a visualization of the flows of program participants through the system, with the width of each of the edges being proportional to the total number of participants moving between states, such as cohort type (Java, data science, UI/UX, etc.); code school provider; whether or not the candidate completed the program; and their ultimate outcome (i.e. terminated or still employed by the firm). Figure 4 shows that there was a roughly even split between RIF eligible and non RIF eligible participants to the program (56% RIF eligible and 44% non RIF eligible). Furthermore, the three largest code schools – Software Guild, Iron Yard and General Assembly had similar graduation rates (12-17%). The Galvanize code school had a 100% graduation rate but only had 5 participants from the firm.

Figure 4 also illustrates how the vast majority of program participants that successfully completed the program ended up remaining at the firm as of 12/1/2019. In contrast, a higher proportion of participants dropped out of the program before graduating ended up being terminated, voluntary or involuntarily, than those who failed to complete the program but managed to remain at the firm. Moreover, it can be seen from the Sankey diagram that most (76%) of all terminations were voluntary rather than involuntary.

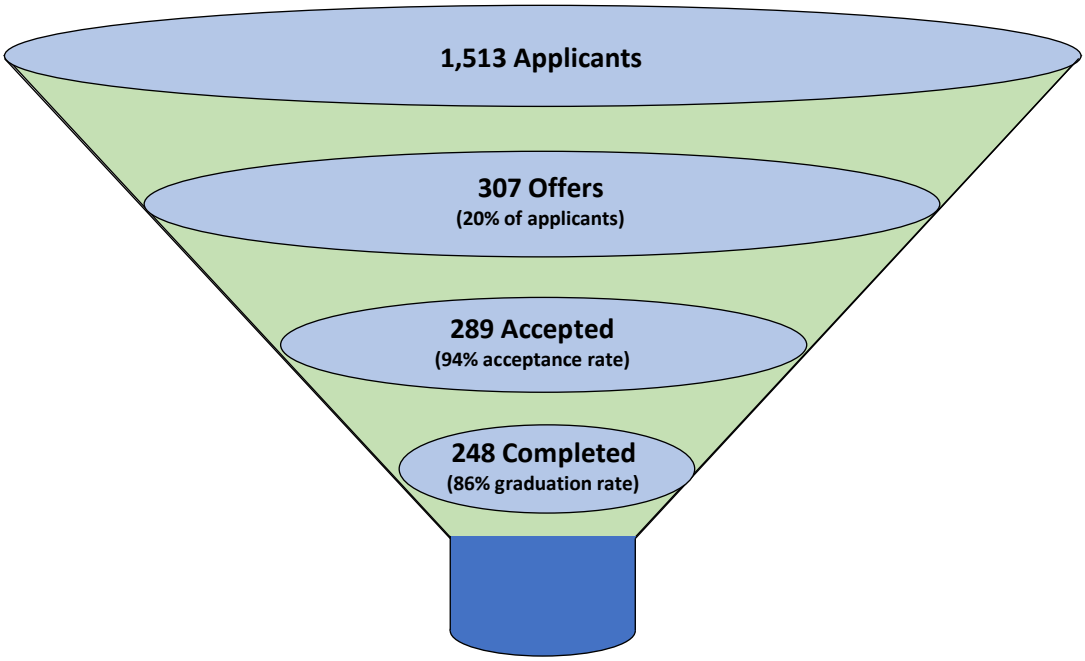

Figure 5: Program recruitment and outcome funnel

Demographic Analysis

The median age of the approximately 1,500 program applicants was 34, with the youngest applicant aged 19 and oldest aged 67. The histogram in Figure 6 plots the age distribution of applicants, with a positive skew indicating that the majority of the applicants were within the 20-40 age range. This can be partly explained by Figure 7, which plots the same data as is shown in Figure 8 but with separate overlaid histograms for RIF-eligible and non-RIF-eligible employees. Here the shape of the plot representing the non-RIF-eligible employees is similar to that of the distribution shown in Figure 5, with a noticeable right-sided skew. In contrast, the histogram for RIF-eligible applicants is more evenly distributed, with a median age of 39, compared to 33 for non-RIF-eligible applicants.

One theory for why non-RIF-eligible applicants tend to be younger than RIF-eligible applicants is that, whether RIF-eligible or not, younger workers are more likely to see value in reskilling opportunities due to being in an earlier phase of their career and have a greater incentive to stay relevant in order to adapt to shifts in workforce skill requirements. Take, for example, the hypothetical case of a 25-year-old Customer Service Representative employed by the firm. Whilst not being directly impacted by the upcoming agile transformation, they may foresee the possibility of their current skillset gradually becoming obsolete (as a result of the increased use of automated customer service chatbots, for instance), and decide to use the program as an opportunity to gain marketable skills that would facilitate a future career change.

In contrast, older employees with a greater amount of work experience and business knowledge are likely to have already found their niche within the firm and consequently feel less

of a compulsion to reskill if their position within the firm is not under threat. Additionally, as program participants were not guaranteed to be given the same job grade or salary upon completion of the program as they had in their pre-program role, it can be argued that younger employees - who were likely to have lower job grades and salaries coming in to the program - had a lower risk of entry. That is to say, older employees that had worked for longer to earn their salary and place within the company, had significantly more to lose from failing to complete the program. It is also possible that older workers decline training because they have a lower time horizon in which to recoup their investment (Dostie, 2015).

The relatively flat age distribution amongst RIF eligible applicants is likely to be due to the firm's shift from Waterfall development to the Agile model affecting a range of different job roles within the company, thus impacting both older and younger employees to the same degree. For instance, both Business Analysts and Project Managers, with median employee ages of 36 and 45 respectively, were both in danger of being displaced by the agile transformation. This likely meant that older employees who were aware that their position at the firm was becoming increasingly untenable, had limited options but to take part in the reskilling program if they wanted to remain employed by the firm.

It can be seen in Figure 8 that the histograms representing accepted and rejected applicants are more or less the same shape, with both curves exhibiting the aforementioned skew towards younger employees. This, along with a median age of 34 for both accepted and rejected program applicants, suggests a lack of ageism within the hiring and selection process for the program.

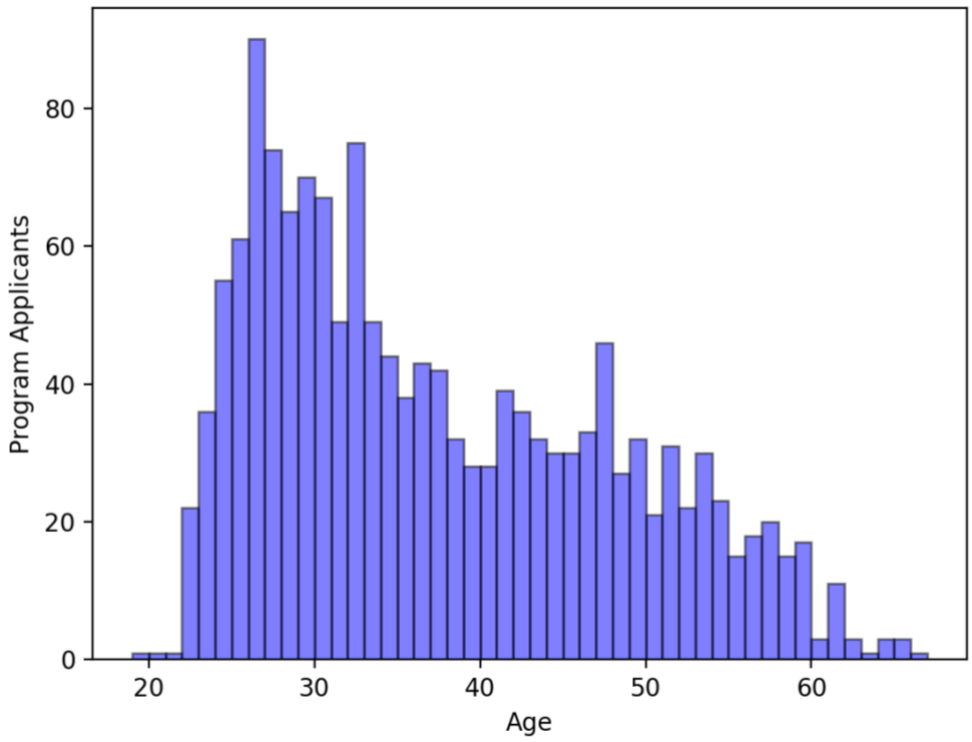

Figure 6: Age distribution of program applicants

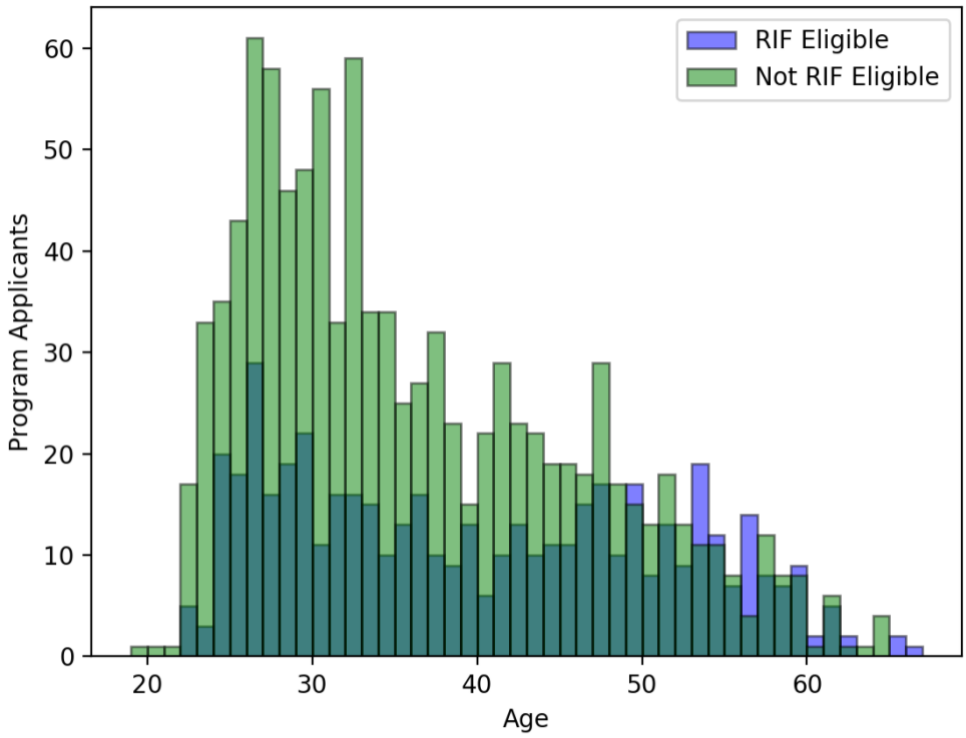

Figure 7: Program applicants by age and RIF eligibility

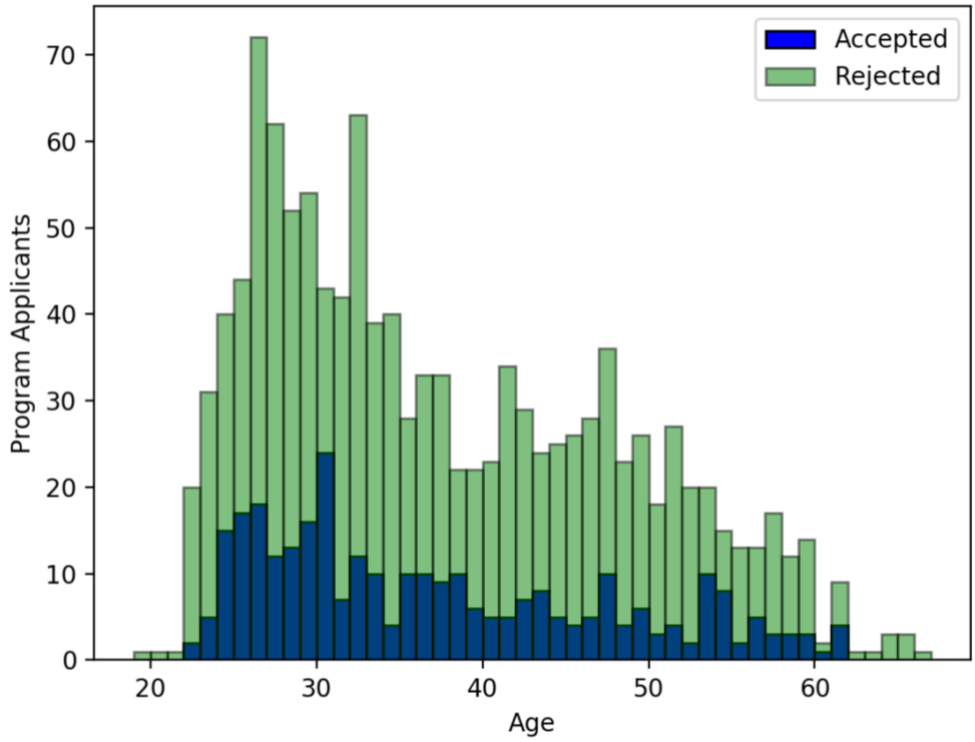

Figure 8: Program applicants by age and application status

Figure 9 shows that female applicants were slightly less successful in earning a place on the program than their male coworkers, with an acceptance rate of 18.0% compared to 21.9% for male applicants. However, Figure 10 indicates that there was a higher proportion of female employees within the pool of program graduates (37.7%) compared to externally hired software engineers (13.4%). In an industry so heavily dominated by men – with women estimated to be occupying only 26% of computing roles in the US (Corbett and Hill, 2015) - it is encouraging to see that the program is providing additional pathways for women to acquire the necessary skills to take up technical roles.

The minority/non-minority split paints a somewhat less rosy picture, however; Figure 11 shows that only 13.4% of minorities applying to the program were accepted, compared to 22.6% of non-minorities. Moreover, minorities constitute 36.0% of all externally hired software engineers but only 16.3% of program participants, as seen in Figure 12, suggesting a relative lack of racial diversity within the program. Figure 13 suggests that the program provided more opportunities for older employees to obtain technical roles, with 26.3% of program graduates being aged 45 or older, compared with 17.8% of external hires.

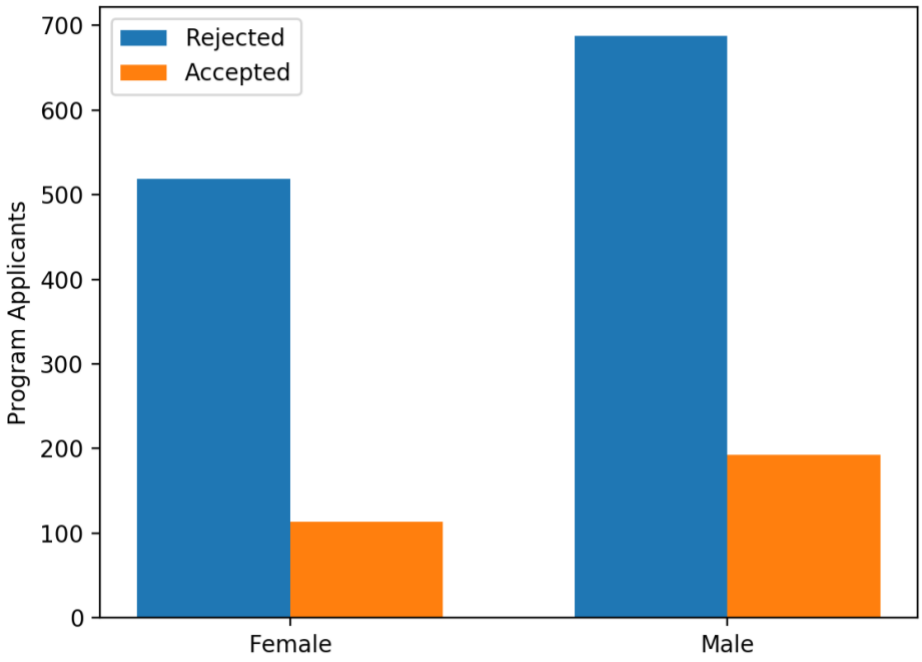

Figure 9: Program applicants by gender and application status

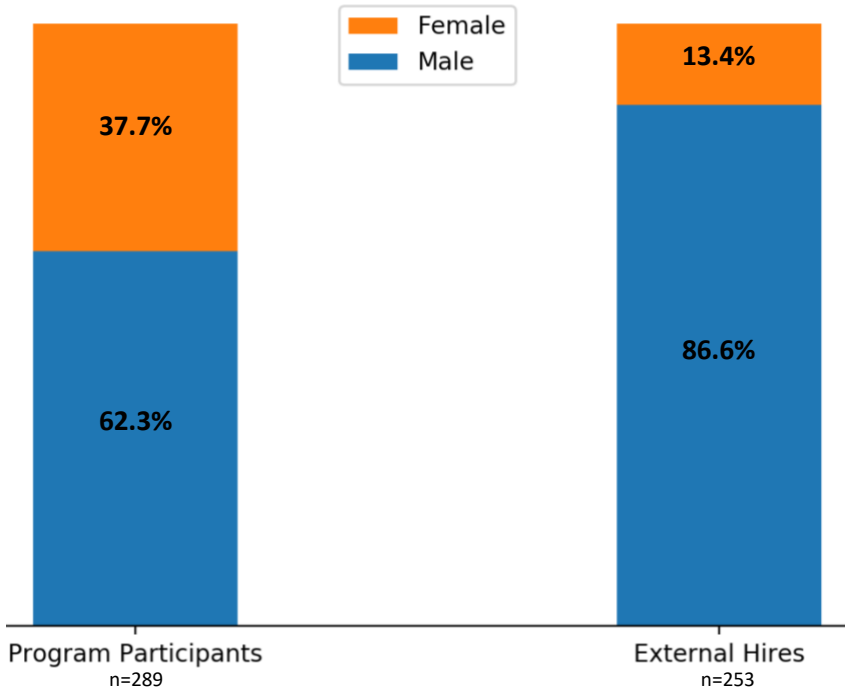

Figure 10: Program participants and externally hired software engineers by gender

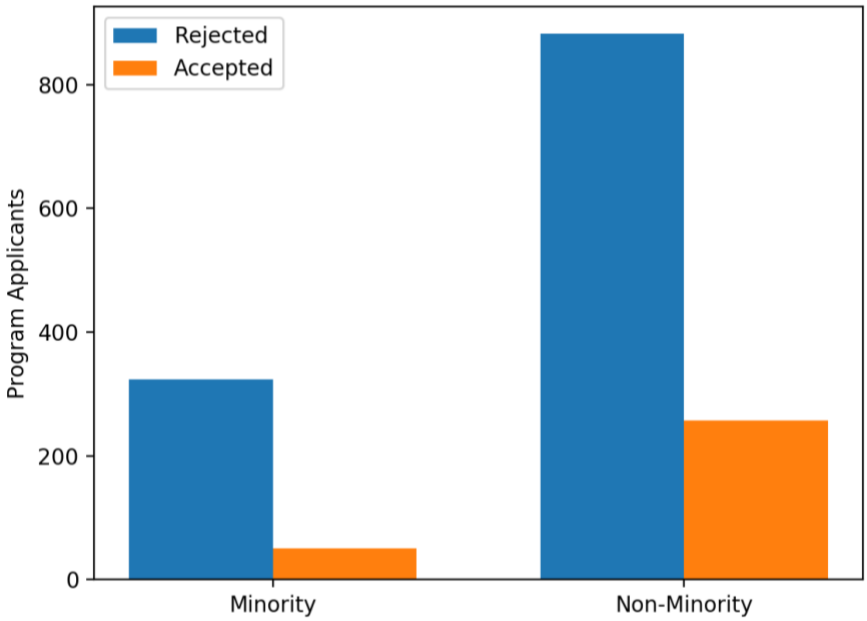

Figure 11: Program applicants by minority status and application status

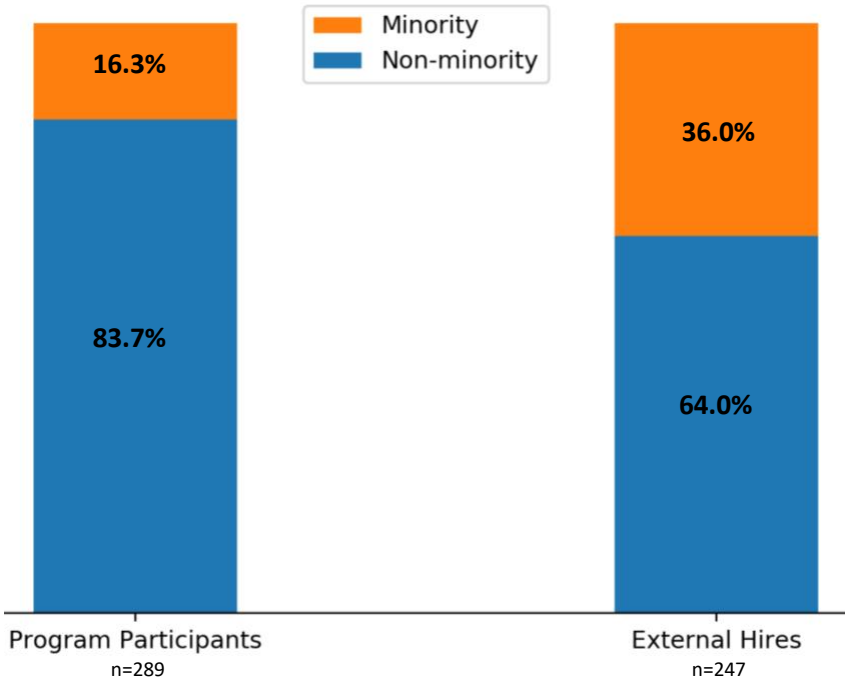

Figure 12: Program participants and externally hired software engineers by minority status

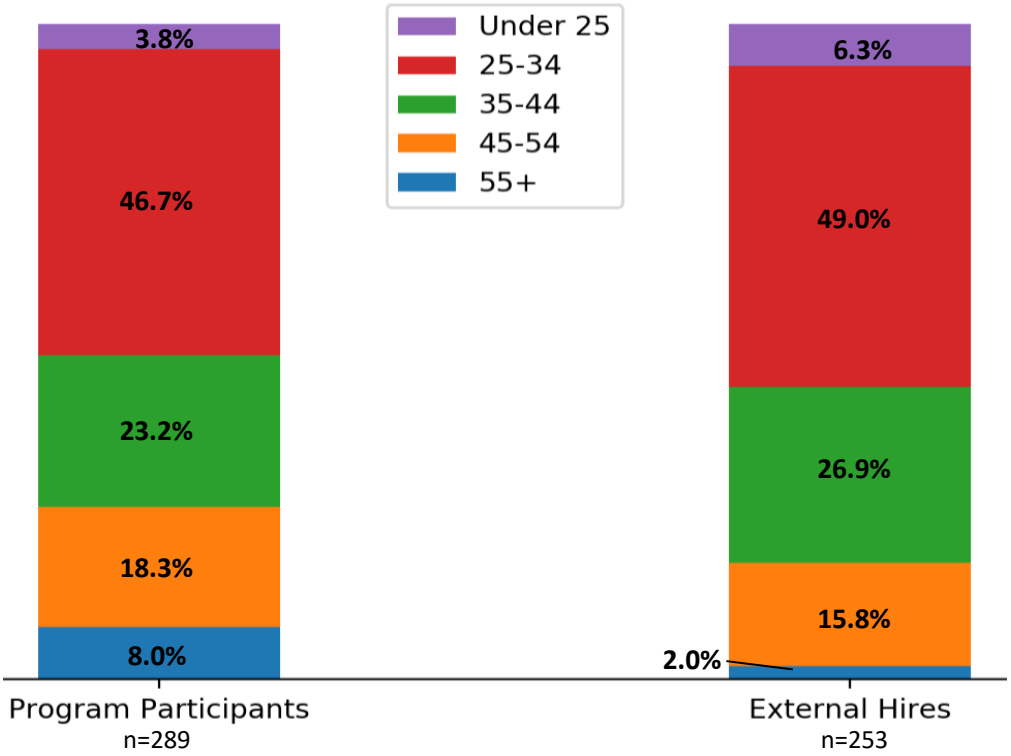

Figure 13: Program participants and externally hired software engineers by age group

Effect of Participant Education Level

Figure 13 shows that the majority of applicants to the program had a bachelor's degree as their highest education level, and that bachelor's and master's degree holders had the greatest success at earning a place on the program, with acceptance rates of 23.2% and 23.7% respectively, whilst applicants with an associate's degree as their highest education level were accepted on to the program 14.6% of the time. None of the 8 doctorate degree holders that applied to the program were selected. There also appears to be little relationship between a participant's highest degree level and their outcome after the program, as shown by Figure 14 where 82.4% of associate's, 86.3% of bachelor's and 81.5% of master's degree holders remained employed by the firm in some capacity as of December 1st 2019.

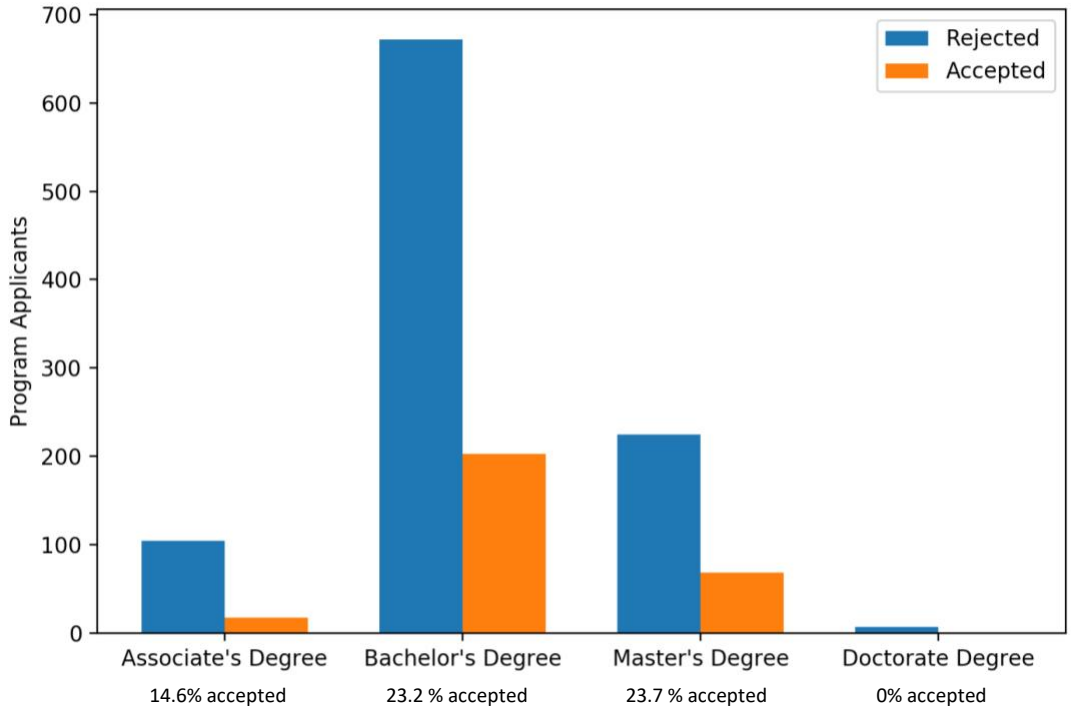

Figure 14: Program applicants by highest education level and program application status

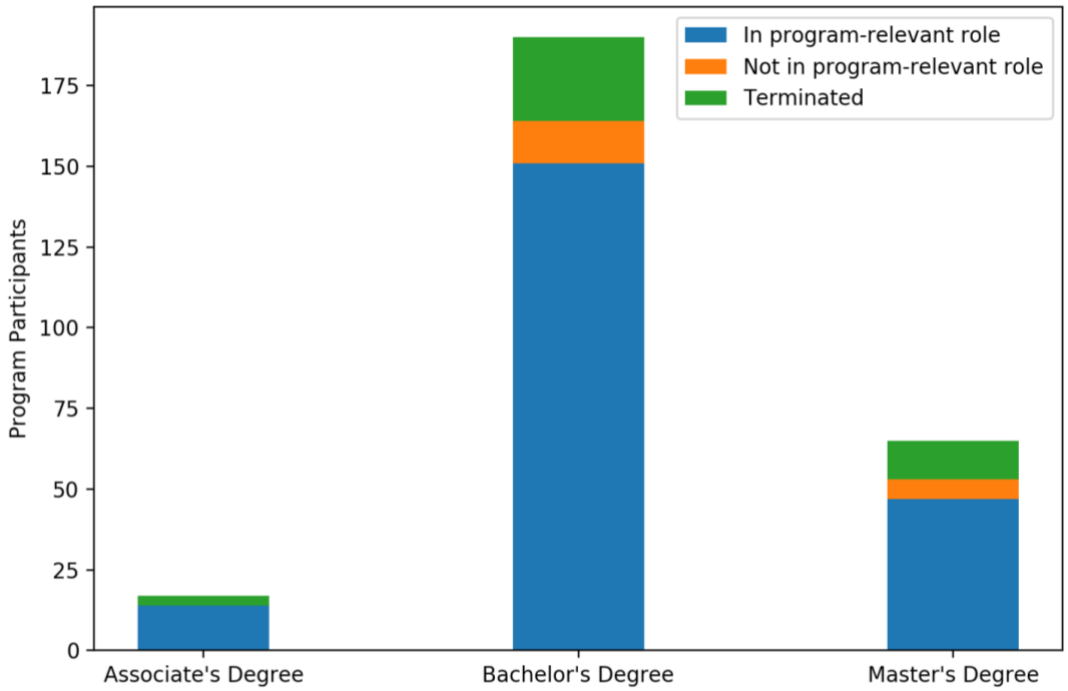

Figure 15: Program participants by highest education level and outcome as of 12/1/2019

Effect of Prior Job Role

Figure 16 lists the 12 job roles from which the program received the most applications, with separate bars for those that were deemed RIF-eligible and those that were not. Most notably, the role of business systems analyst had the highest number of applications with 335, almost double that of the second highest role, customer service representatives, with 174. Business systems analysts also had by far the highest proportion of RIF eligible applicants at 92.2%. Other roles with high levels of RIF eligibility were systems analysts and project managers at 70.2% and 62.9%, respectively. As previously mentioned, all three of these positions were particularly at-risk due to being gradually phased out by the firm's transition from Waterfall to Agile software development. The high volume of applications to the program from employees in these roles can therefore be largely attributed to them viewing it as an opportunity to survive the agile transformation and remain employed by the firm.

The high volume of applications from customer service representative and claims specialist roles is most likely a result of there simply being a high of number people employed within these roles at the company, with a total of 7,191 Claims Specialists and 3,284 Customer Service Representatives as of 1/31/2017, compared with only 1,205 Business Systems Analysts. Figure 17 presents the twelve job roles with the highest number of applicants to the program per 100 employees; any job roles with less than 100 total employees at the firm were omitted from the plot. Customer Service Representatives are ranked as having only the ninth highest rate of applications to the program, with 5.3 applications per 100 employees. Claims Specialists averaged 2.27

applications per 100 employees and do not make it in to the top 12. In contrast, roles that are shown in Figure 16 to have to have some of the highest levels of RIF eligibility, Business Systems Analysts and Systems Analysts, also appear to have the highest rates of application to the program in Figure 17.

The same 12 job roles in Figure 16 are once again presented in Figure 18, but this time with each role's acceptance rate for the program. It is interesting to observe that the job roles with highest RIF eligibility shown in Figure 16, namely Business Systems Analysts and Systems Analysts, had relatively high levels of success at being accepted on to the program. A possible explanation for this is that employees within these roles, whilst not expected to be able to code, are more likely to require a higher level of technical capabilities to do their job compared to applicants coming from most other job roles. For instance, in order to successfully analyze business processes, Business Systems Analysts will often need to employ techniques such as data modelling, requiring the ability to manipulate spreadsheets and manage databases through the use of Structured Query Language (SQL). These technical skills are more likely to lend themselves towards learning how to code and may therefore have given these individuals an edge over applicants from roles such as Customer Service Representative and Claims Specialist, who would not have required quite the same level of technical ability to fulfil the job requirements.

Not surprisingly, the role of software engineer was the role included in Figure 16 and Figure 18 that exhibited low rates of RIF eligibility and high rates of admittance on to the program. As the role of software engineer requires high levels of technical ability – most crucially, the ability to code using computer programming languages – this provides further evidence that technical skill proficiency was a determining factor in an applicant's chances of being accepted on the program

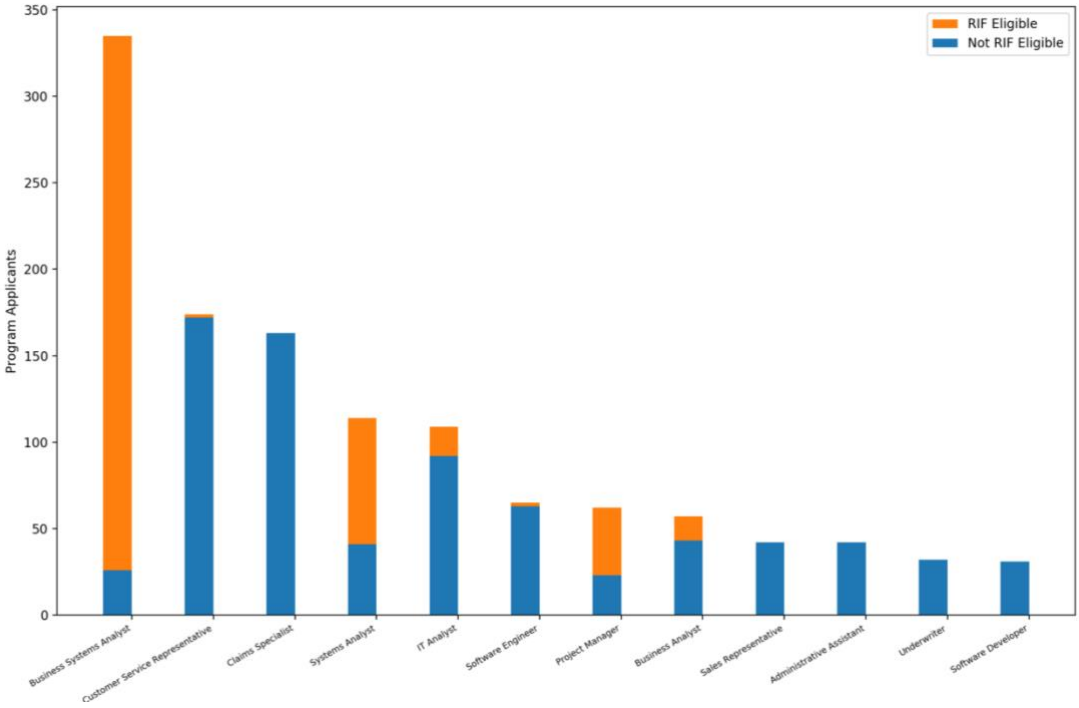

Figure 16: Program applicants by pre-program job title and RIF eligibility

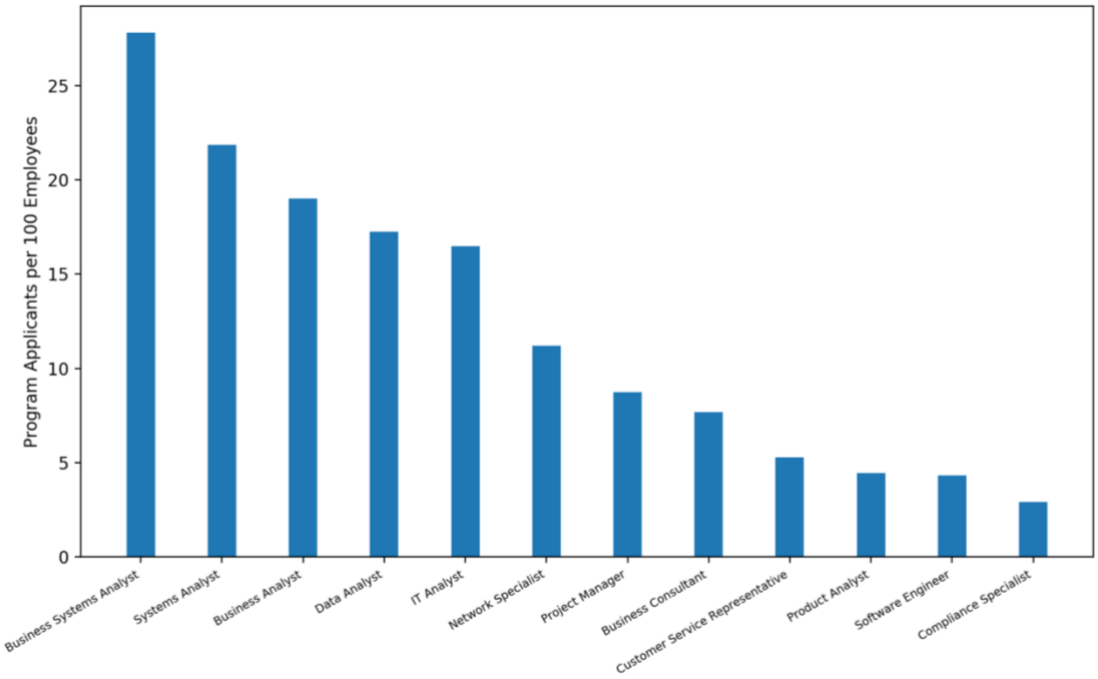

Figure 17: Job titles with the highest number of program applicants per 100 employees (job titles with headcounts less than 100 employees are omitted)

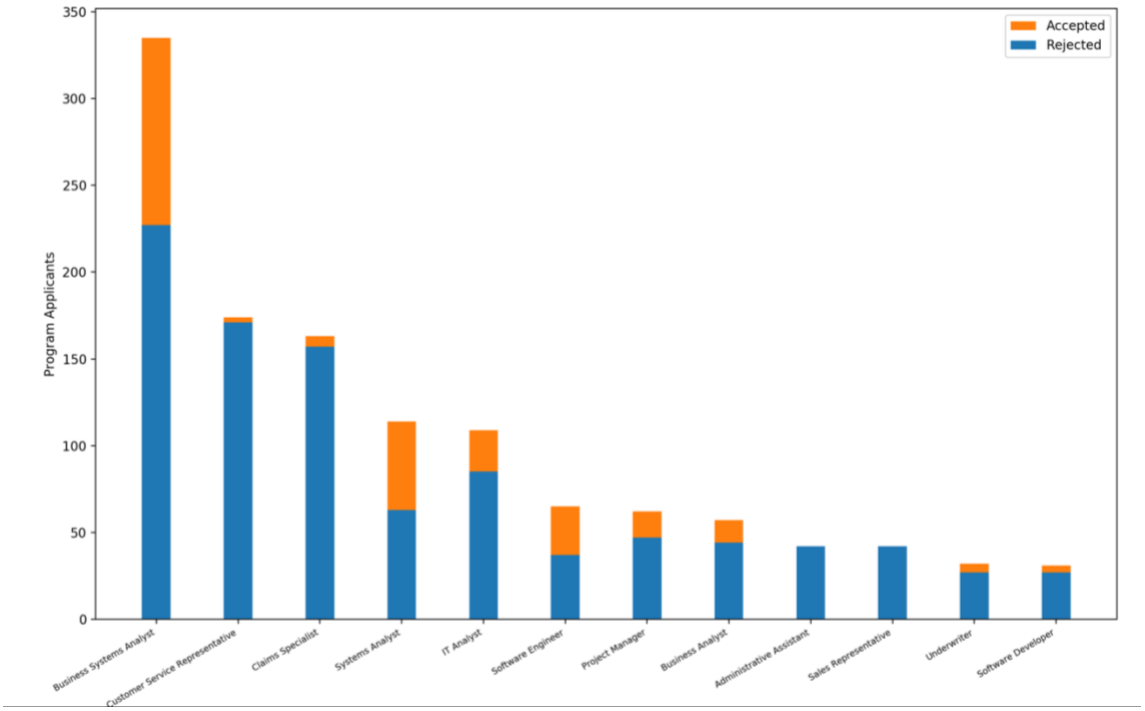

Figure 18: Program applicants by pre-program job role and program application status

Effect of Prior Coding Experience

Pre-program coding experience appears to be highly indicative of a participant's success on the program. Here success is measured by a participant's post-program job grade, which was assigned nine months after completion of the code school, and whether or not they remained at the firm in a program-relevant role as of 12/1/2019. The mean post-program job grade for all program participants with prior coding experience was 14.3, with a standard deviation of 1.04. For participants without prior coding experience, the mean job grade was 13.9, with standard deviation of 0.86. This suggests that participants with prior coding experience reached, on average, a higher level within the company upon completion of the program than those with no prior coding experience.

Furthermore, Figure 19 shows that 76.4% of program participants with prior coding experience remained in program-related roles 9 months after the code school, compared to only 45.8% for participants without any prior coding experience. This may have been due to program participants with prior coding experience having had a head-start over those whom were completely new to coding, allowing them to reach a higher level of technical proficiency upon completion of the program and thus outcompete those without prior coding experience. Another potential reason is that participants whom had previously already learnt to code were likely to have a greater interest in software development and would have made an effort to remain within a technical role after the program ended. In contrast, some participants with no prior knowledge of coding may have realized, through taking part in the program, that they did not have a sufficient

interest in software development to pursue a career in it, and consequently decided to apply for a less technical role within the firm instead.

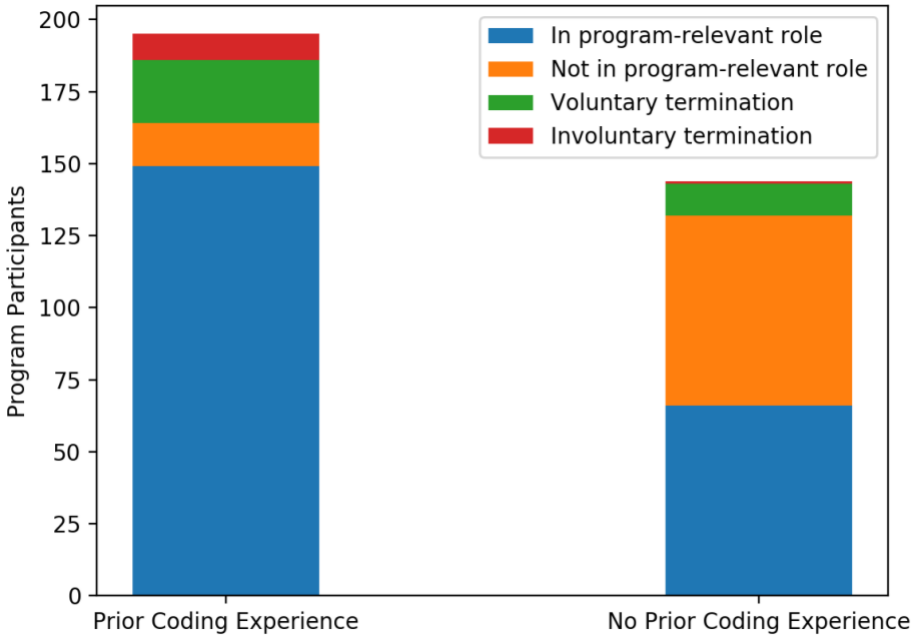

Figure 19: Program participants by prior coding experience and outcome as of 12/1/2019

Outcomes of RIF-Eligible Employees

One of the key components of this study was to assess the impact that the reskilling program had on employees that were in danger of being let go by the firm. It is encouraging to see then, as evidenced by Figure 20, that RIF-eligible employees were well represented amongst program participants. Here we can see that of the 499 RIF-eligible employees applying to the program, 170 were offered a place, giving an acceptance rate of 34.0% compared to only 13.5% for the 1,014 non-RIF-eligible employees that applied to the program. As previously discussed, this is probably due to employees in highly RIF eligible positions such as business system analysts and system analysts having more advanced and transferable technical capabilities than Customer Service Representatives and Claims Specialists.

Another possible factor is that applicants in danger of losing their place within the firm had a greater incentive to prepare thoroughly for the interview and consequently performed better. There is also the possibility that the selection process was in some way biased in favor RIF eligible employees over non RIF eligible employees; however, this seems unlikely due to interviewers for the program revealing that applicants who listed a desire to save their job as one of their primary reasons for applying to the program were not looked upon favorably.

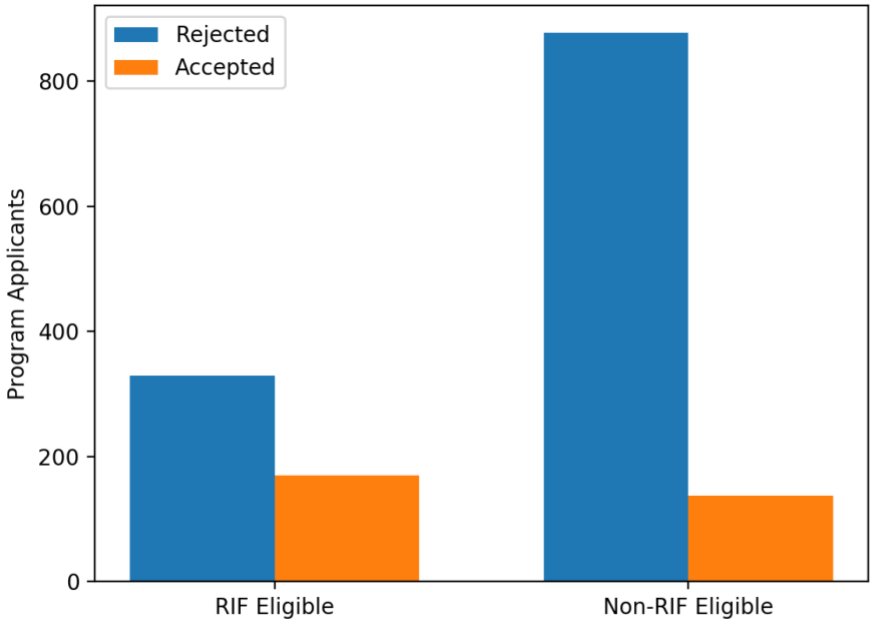

Figure 20: Program applicants by RIF eligibility application status

Whilst we have seen RIF-eligible employees were disproportionately successful at being accepted on to the program, when it comes to program outcomes however, it appears that RIF-eligible applicants did not perform as well as their non-RIF-Eligible co-workers. RIF-eligible program participants achieved a mean post-program job grade of 13.97 (with standard deviation of 0.74) compared to 14.49 (with standard deviation of 1.19) for non-RIF-eligible participants. This is illustrated in Figure 21, which compares the number of RIF-eligible and non-RIF-eligible participants that were assigned each post-program job grade. Here it can be clearly seen that almost four times as many non-RIF-eligible employees earned the highest job grade of 16 than RIF-eligible employees. Non-RIF-eligible participants are also more likely to remain in a program-related role after completion of the program, as shown in Figure 22 where 85% of non-RIF-eligible participants ended up in a program-related role, compared with only 70% of RIF-eligible participants.

Evidence from qualitative interviews may provide an explanation for why non-RIF-eligible employees have, on average, better post-program outcomes than RIF-eligible employees. Due to the challenging nature of the code school and job requirements of a software engineer, a large number of participants and managers noted that, in order to succeed on the program, participants had to have a high level of mental fortitude and a strong desire to grasp the technical concepts being taught. Even after the code school, when participants had been placed into technical roles within the company, they were expected to be highly independent and continuously teach themselves new concepts due to the rapidly evolving software landscape. Consequently, the individuals that exceled were the ones that were strongly committed to a career within the field of software development and had a strong interest in learning how to code. In contrast, participants

that applied to the program with the primary objective of saving their place within the firm, due to being RIF eligible, were less likely to achieve equivalent levels of success.

The Sankey diagram presented in Figure 25 illustrates the outcomes of all employees within the firm that were deemed RIF-eligible during 2016-18, regardless of whether or not they applied to the program. It is interesting to observe that only 23.4% of RIF-eligible employees decided to apply for the program, whereas 74.3% applied for a different role within the firm, either in addition to or instead of the reskilling program. This suggests that a large proportion of people simply have no interest in in learning how to code and transitioning to a technical role, even if their current job is at risk. As an accompaniment to the visualization shown in Figure 25, Table 2 provides a percentage breakdown for the final outcomes of employees that either applied to a different role in the company (including employees that were rejected from the reskilling program) or applied to neither the program nor a different role in the company.

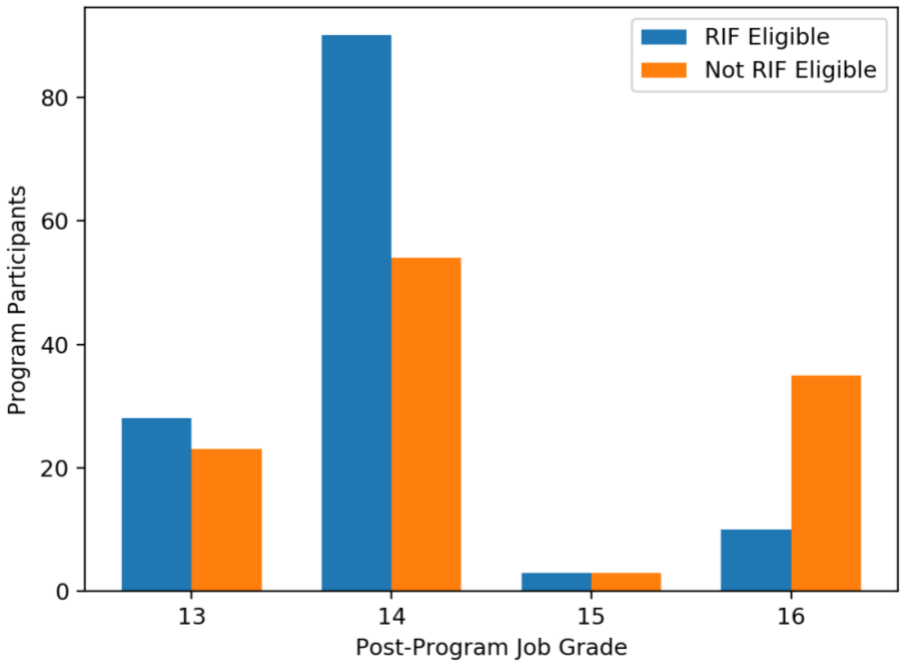

Figure 21: Program participants by post-program job grade and RIF eligibility

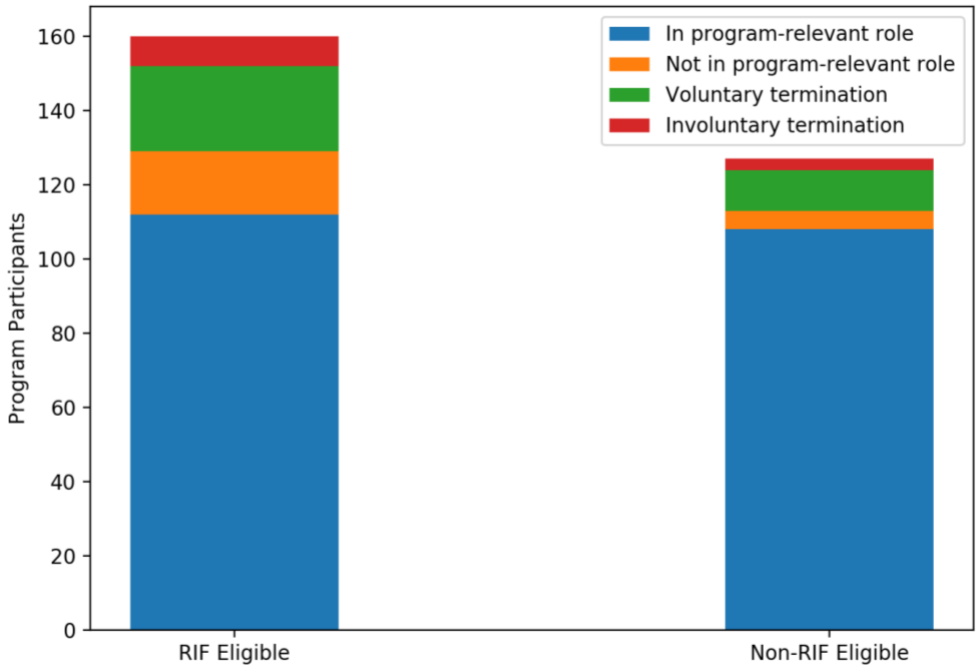

Figure 22: Program participants by RIF eligibility and outcome as of 12/1/2019

	All RIF-eligible employees	***Applied for a different role at the company***	***Applied for neither the program nor a different role***
Not Terminated	1308 (61%)	669 (65%)	454 (56%)
Voluntary Termination	248 (12%)	91 (9%)	121 (15%)
Involuntary Termination	576 (27%)	269 (26%)	233 (29%
Other Termination	2 (0%)	1 (0%)	1 (0%)
Total	**2134**	**1030**	**809**

Table 2: Summary of RIF eligible employee outcomes

Comparisons with external hires

In addition to the aforementioned Figure 10 and Figure 12, which compare gender and minority status splits for both program graduates and external hires, the firm also carried out some independent analysis on the performance and eventual outcomes of program graduates when compared to externally hired software engineers. Figure 23 shows that 90% of program graduates remained employed at the firm 18 months after beginning the program, whereas only 80% of externally hired software engineers were still employed by the firm 18 months from the date they were hired.

Figure 24 compares the proportion of program participants that remain in program-relevant roles, whether at the firm or elsewhere. The job type of ex-employees was ascertained through looking them up on LinkedIn, hence the small sample size of 29. We can see that, of the program participants that were still at the firm as of 4/1/2019, 91% percent remain in a program-relevant role, compared to 41.4% of participants that left the company.

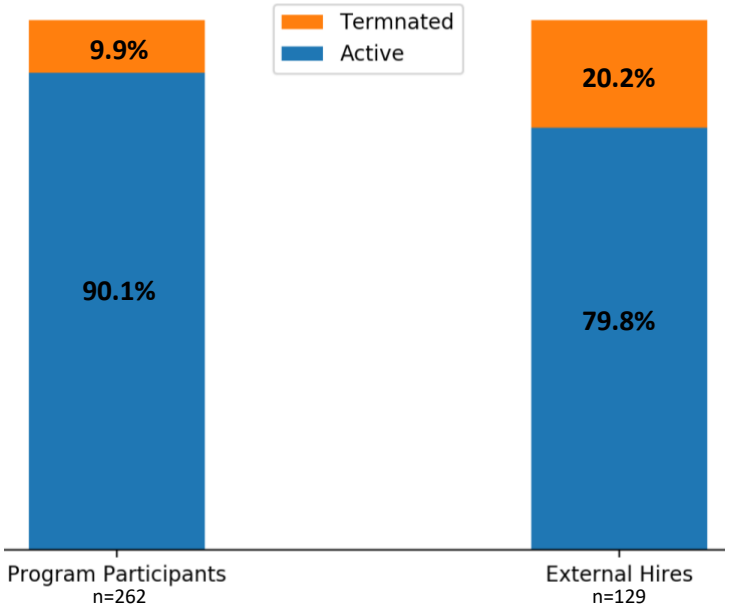

Figure 23: Program participants and externally hired software engineers by employment status after 18 months of program start date or hire date (includes all software engineers externally hired during the year 2017)

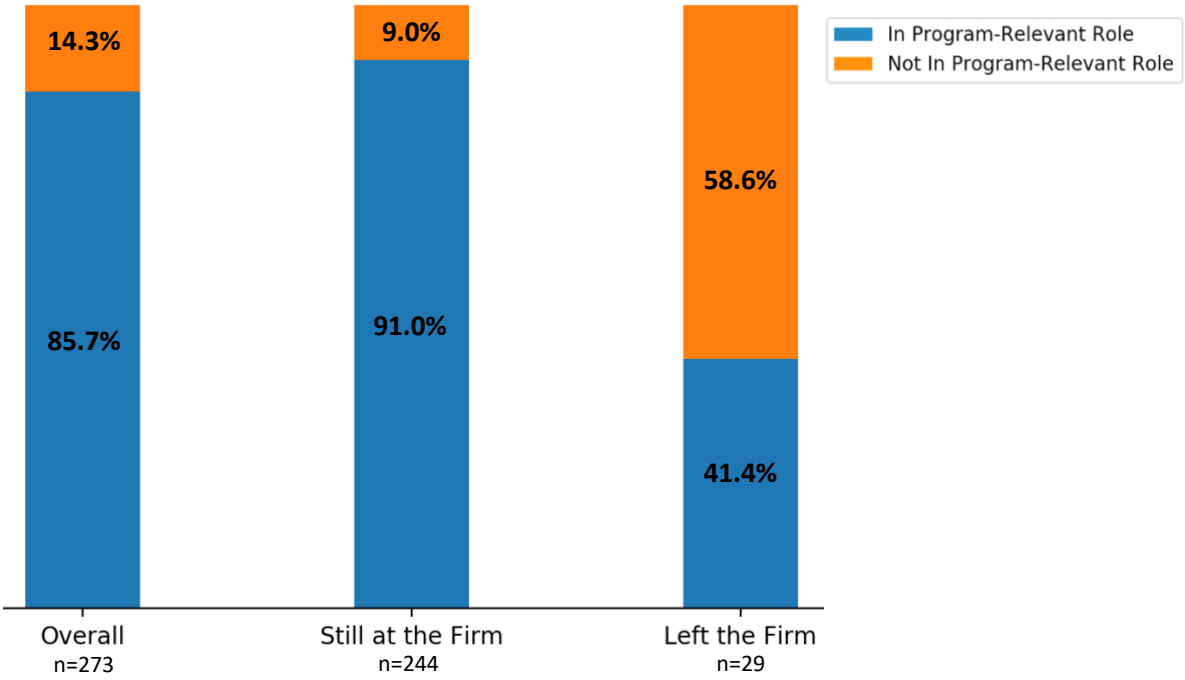

Figure 24: Likelihood of program participants still being in a program-relevant role after having stayed at or left the firm

Cost of running the program

Running the code school entailed a significant investment by the firm as they were required to pay for the costs of tuition, travel and accommodation expenses, in addition to maintaining participant's salaries whilst they attended the code school. However, through hiring internally, the firm would also have made cost savings through not having to pay recruitment costs for new hires and avoiding severance payments when terminating RIF-eligible employees.

We were not provided with detailed financial data as part of this study. However, the firm conducted analysis of their own in order to calculate the cost per year required to put a single employee through the program, with the net investment amortized over the participant's length of service after completing the program. Their analysis found that the yearly cost of putting a participant through the program was significantly affected by whether or not the participant was due to be terminated by the firm. RIF-eligible participants cost the firm on average $2,596 a year, whereas putting a RIF-eligible employee through the program – instead of terminating them – ended up saving the firm $37 a year, as it meant that they avoided having to pay severance costs.

In summary, the analysis provided in this chapter involved comparing program applicants based on numerous demographic variables; measuring the effect of participants' prior job role, education level and prior coding experience; assessing the outcomes of RIF-eligible employees; comparing the outcomes of program participants with the outcomes of external hires; and analyzing the cost of running the program. These findings will be used to support the qualitative

analysis provided in the following chapter and form the basis of many of the concluding remarks in Chapter 7.

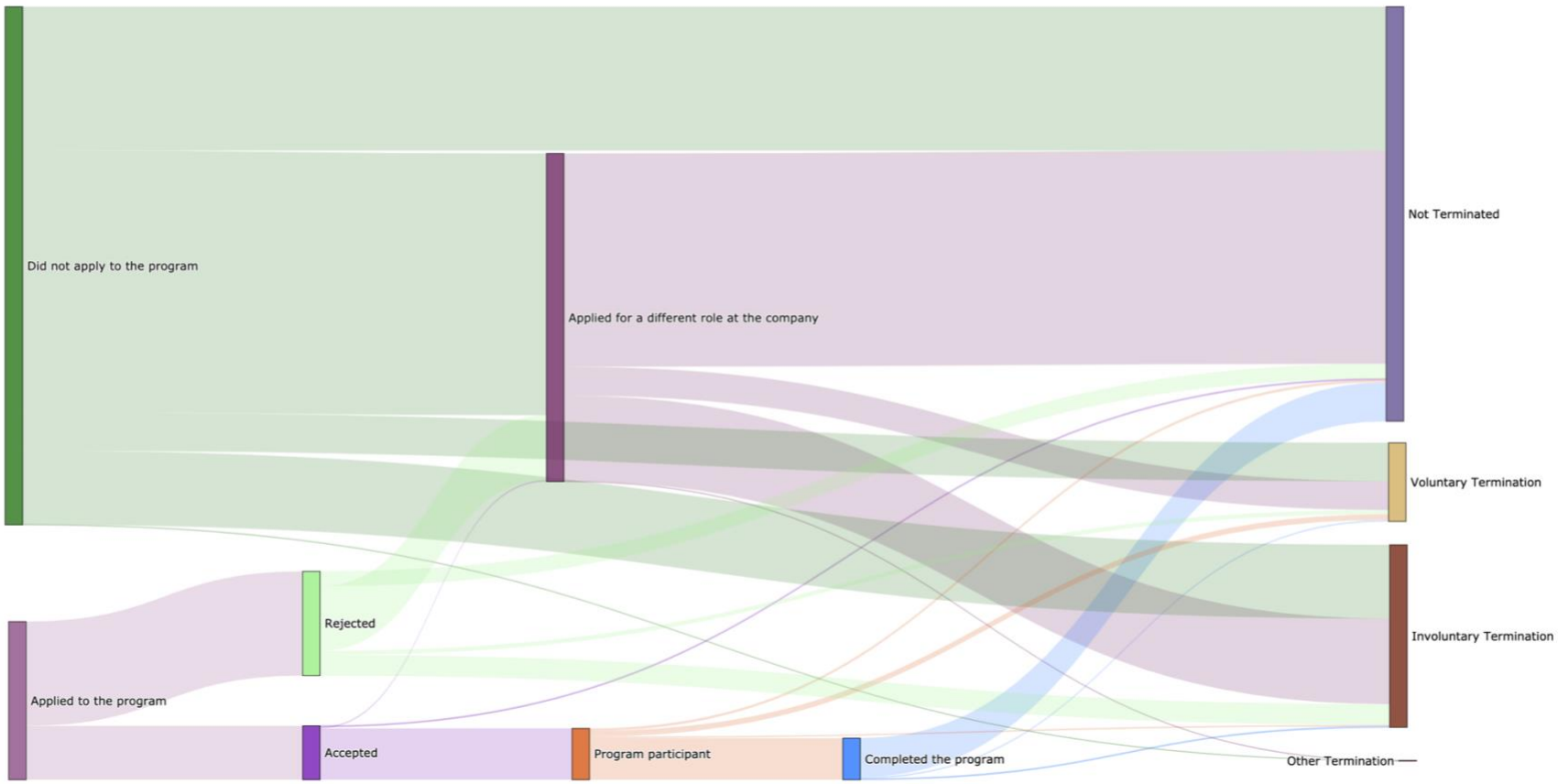

Figure 25: Sankey Diagram of outcomes for all RIF eligible employees

Chapter 6: The Effects of Reskilling on the Firm and the Worker

Retention of Business Knowledge

Perhaps the greatest triumph of the program from the firm's perspective was the retention of business- and industry-specific knowledge within the company. As discussed in Chapter 3, in the days of the Waterfall development model business systems analysts acted as a bridge between departments to help IT members understand business requirements and help management professionals understand the technical software solutions, thus working as a "tech-translator" of sorts. By training these individuals to be software engineers and placing them directly into technical teams, they were able to fulfil the same role whilst also adding technical value. Program participants reported that they were able to act as a "middleman" between the business and technology arms of the company by taking their firm-specific knowledge (such as an understanding of company practices and familiarity with coworkers) in addition to an awareness of the nuances of the insurance industry, and applying them within their new roles. As mentioned by one participant:

> *"A lot of the web engineers I've worked with know a lot of tech and how to create software, but they don't know much about the business of claims. I think I've been very helpful in knowing what's involved in a claim, and what do customers need."*

Having an all-round knowledge of both the business context and the technical details is extremely valuable within Agile teams, where seamless communication between business experts and technologists is a vital component of the development workflow. Software engineers that had graduated from the program were able to easily comprehend the user requirements for a proposed software change and then communicate this back to other members of their team in a way that would make sense to them by translating it into software engineering terminology. Debugging was also made easier as developers with an in-depth knowledge of what the functionality of a business application should be generally found it easier to diagnose a problem within the code, thus saving a considerable amount of time during the testing and development phases.

Keeping business-specific knowledge in-house and preserving social capital among current employees was clearly an intended outcome for the firm as efforts were made to assign program graduates on to teams in which they were likely to contribute more than just technical expertise. Many of the program participants that were interviewed as part of this study were intentionally placed on teams in an areas of the firm where they had previously worked and were familiar with, so that they could provide valuable business knowledge during the software engineering process. For example, if a program participant had been working as a sales representative prior to taking part in the program, they would most likely be placed on a development team that was in charge of designing software for the sales teams within the firm. Having previously worked within sales,

the program graduate would know more or less exactly what the end users wanted from the application, thus allowing for time to be saved during the requirements stage of the software development cycle.

Being able to contribute business-specific knowledge to the team workflow was also an important morale boost for code school graduates who were suffering from imposter syndrome and felt as though they lagged behind externally hired developers in terms of technical capabilities. Despite generally having significantly more experience than program graduates, externally hired software engineers rarely have a sufficiently detailed knowledge of the insurance industry to effectively liaise with insurance experts such as underwriters. During qualitative interviews, one technology manager argued that, as the insurance industry is extremely complex, teaching a software engineer to understand the intricacies of the industry can be equally as challenging and time-consuming as teaching someone how to code. In this regard, the reskilled workers are therefore likely to bring more to the table than externally hired software engineers with no prior knowledge of the business or insurance industry. Consequently, even when program graduates were struggling to keep up with their team members when it came to developing software, they still felt as though they were able to provide insights that more experienced software engineers were unable to. From the firm's perspective, investing in these employees and trying to retain them, rather than letting them go and trying to hire from outside the firm, made much more sense.

The overall effect of retaining business knowledge on both productivity of the agile teams and the morale of the employee was nicely summarized by one program graduate:

> *"It happened that the group I ended up on had very little understanding of the core fundamentals of insurance, and the process and procedures of underwriting departments. So my software development has been slow, but my ability to communicate and help the team focus on what really needs to be done. I can look at inbound bug fixes to figure out the user request and translate that into engineering terminology, and vice versa. This has really helped make my team's work more concise and on-point […]When you come out of a cohort, you know it'll take a while before you're on par with people who have a 4 or 6 year degree– so it's good that I had something to contribute immediately."*

Job Retention

Needless to say, the implementation of the program provided significant benefits for the majority of participants that took part. Perhaps most crucially, it provided an opportunity for employees whose jobs were under threat to remain employed by the company. Of the 248 employees that successfully completed the program between 2016 and 2018, 131 (or 52%) of them were classified as RIF eligible. The program therefore acted as a lifeline to these at-risk employees, allowing them to reskill in order to remain useful to the firm and keep their place at the company. Whilst it is likely that young employees would have gone on to find new roles in a different firm, some older employees that we interviewed admitted that they likely would have been forced into early

retirement if let go by the firm. This supports the view that positive reciprocity from employer-provided reskilling results in a delayed retirement age and prolonged working life for older workers (De Grip *et al.*, 2019), thus benefitting both the firm and the reskilled employee.

The program also served as a gateway in to technical and coding-based roles for many people who otherwise would not have had the opportunity to make this transition. Qualitative interviews with program participants and technology managers revealed that the program provided a "second chance" for employees to pivot in their careers and move in to a new field of work within the technology industry, particularly for those without a degree in a STEM-based subject or any prior technical experience. If not for the program, individuals looking to move into a software engineering role would have had to either enroll at a university or independently apply to a code school, which would likely have cost a significant amount of money and also required them to take time off work in order to go through this reskilling process. Thus, through providing the reskilling program free of charge, the firm allowed their employees to reskill with no financial burden whilst also allowing them to keep their salary during the code school.

Through reskilling, the program also provided new career advancement opportunities for employees that were either dissatisfied with their previous roles at the company or felt as though they had hit a "glass ceiling" in terms of their career trajectory. Many of the program participants that were interviewed as part of this study spoke of how they were using their newly-acquired knowledge and skills from the program as a platform for moving into specialized technical areas such as software architecture or data science. Other participants that were able to leverage both their knowledge of the insurance industry and their technical expertise found themselves ideally placed to pursue technology manager roles, such as product owner or scrum master, within the new Agile model. Thus, the program was successful in significantly increasing the value of participants not only within the firm, but also within the external marketplace, thus allowing them to negotiate higher salaries (Haelermans and Borghans, 2012).

Impact of the Program on RIF Eligible Employees

The quantitative analysis provided in the previous chapter identified a clear disparity between the performance of RIF-eligible program participants and that of non-RIF-eligible participants. Those who were RIF-eligible achieved a mean post-program job grade of 13.97 (with standard deviation of 0.74) compared to 14.49 (with standard deviation of 1.19) for non-RIF-eligible participants. Furthermore, 85% of non-RIF-eligible participants ended up in a program-related role, compared to only 70% of RIF-eligible participants. As discussed, this may have a lot to do with non-RIF-eligible participants displaying a greater willingness to partake in the program due to having a genuine interest in learning how to code, rather than choosing to apply in order to avoid being terminated by the firm, as may have been the case for a number of RIF-eligible participants.

This finding suggests the existence of a trade-off when it comes to selecting candidates to take part in the program; that is, applicants to the program who are at risk of being let go by the firm may not necessarily be the same applicants that have a strong desire to reskill and consequently succeed on the program. The firm may therefore be left with a difficult decision on

their hands: do they choose to prioritize employees that are in danger of losing their jobs, thus allowing them to remain at the firm, or do they instead favor candidates that show the greatest potential to thrive in a technical role? As discussed in the previous chapter, successfully reskilling a RIF-eligible employee instead of a non-RIF-eligible employee could save the firm on average $2,633 a year, suggesting that the firm has a financial incentive to favor RIF-eligible applicants. However, these cost savings could easily be outweighed by losses incurred through selecting RIF-eligible applicants who are not suited for the program and consequently drop out.

Even in situations where the primary objective of the firm is to reskill and reassign as many employees as possible, regardless of prior technical experience or interest in learning how to code, this may not always result in the best outcomes for some workers, as the majority of participants that ended up dropping out of the program did so as a result of lacking either an aptitude for coding or the enthusiasm for pursuing a career in technology. Qualitative interviews with program graduates and technology managers suggest that, whilst the ability to code is certainly something that can be taught, it is still to some extent an innate skill necessitating a high degree of logical cognition and the ability to problem solve. According to one program graduate:

> *"They say anyone can do coding– but now I've done this for 3 years, it's not as simple as it sounds. You can learn coding, but I don't think it's for everybody. Even if there were better support, I think some people would still have struggled. One good friend in particular [...] she was one of the business systems analysts affected by job cuts, and she got into [the program] but she really, really struggled in thinking like a programmer. She ended up quitting and getting a product owner role in another company. [The program] helped her think about what else she could do in tech, not just as a developer. These are smart people, but not everyone can do coding and love it."*

Qualitative interviews also consistently emphasized that successful coders must have a strong desire to continuously learn and refine their trade in order to keep pace with the industry. One senior technology manager, who had supervised a number of program graduates, said the following:

> *"I think when it comes to being successful as a software developer, a lot of people need to have it as a side passion rather than just doing it during the day. Some folks are able to be successful just having it as a day job, but I would say most people that are successful at it like reading about it and are constantly learning because the landscape changes so quickly and is very competitive."*

Consequently, it is vital that candidates who do not have any prior technical experience related to coding, or do not exhibit sufficient logical or problem-solving capabilities, have a strong passion for technology or else they are unlikely to succeed on such a program. Whilst it may seem like the most noble course of action would be to allow every member of the company to have access to this reskilling opportunity, situations where participants are forced to drop out of the code school have negative implications for both the participant, who has gone through the trauma of failing the

code school and is now left without a job, and the firm, having wasted time and money on training the employee.

Onboarding and Integration Issues

Qualitative interviews with participants and managers uncovered a number of issues concerning the onboarding process, which help to explain the paltry average onboarding satisfaction score across each of the 18 cohorts as seen in Table 1. The firm aimed to assign participants on to teams early whilst they were still at code school or soon thereafter. However, frequent reorganizations of departments and development teams, or the cancellation of large projects that occurred in the three months whilst the code school was being implemented, meant that it was not always possible to find vacancies on teams for all program participants by the time they completed the code school.

As a result, many participants experienced long delays in being placed on to a team, with some being forced to wait up to a year before eventually being assigned a new role within the firm. During this time, unassigned employees either switched between teams, working on separate projects at a time, or were given side-projects to work on independently. Qualitative interviews revealed that participants who were subjected to long delays during the onboarding process experienced a considerable amount perturbation due to the uncertainty around their future, in addition to a sense of frustration at not being able to properly put their new technical skills to use. Unassigned participants also complained about experiencing skill atrophy as a result of having insufficient opportunities to apply their newly acquired skills.

> *"It took people a while to get placed onto teams because they didn't anticipate such a hard matching process after the program. I think some took almost a year, maybe a little less to be put on teams. They were put on odd projects here and there but it was an uncomfortable time for them. They had all these skills but nothing to do with them."*

In other instances, program participants were placed on teams and projects that did not match apply the skills and knowledge that they had obtained from the code school. Some participants complained of not being given enough technical or coding work, and instead were asked primarily to perform tasks such as writing requirement documents for the team, whilst others were required to perform tasks for which they were not trained for. This most commonly occurred to participants that went through the front-end development cohorts but ended up being assigned to back-end teams. Naturally, many of the skills that are required of a back-end developer, such as the ability to interface with databases and servers as well as the ability to code in back-end languages such as Java or Python, would not have been taught in much detail, if at all, within the front-end code school. Those that were unable to quickly learn back-end development on the job either decided to leave the firm or find alternative roles within the company.

> *"Some people unfortunately ended up just doing backend, maybe because a big front end project got cancelled and there were only a few spots left around the area after that. Bad timing overall, it was a little rocky. What was the point of all that front-end work [on the program]?"*

Perhaps the most egregious case of the program onboarding process going wrong involved the single data science cohort that was run as a pilot program at the end of 2016 with only five participants in total. After completing the code school, participants from this cohort were effectively shunned by their newly assigned teams as a result of being considered not up to an acceptable standard, especially when compared to externally hired data scientists within the firm that had graduate-level qualifications in data science or statistics. None of the graduates from this cohort ended up in data science roles and instead were hastily reassigned to other positions such as software engineer or testing engineer. The fact that the firm decided not to run any subsequent data science cohorts after this initial pilot indicates that they recognized the failure of the program in this instance, suggesting that a 3 month code school is an insufficient amount of time to train an individual to be an industry-level data scientist. This reality is perhaps underscored by the significant demand for data scientists in the labor market.

As the first nine months after completing the code school are crucial for determining a participant's job grade and salary after the program, these instances of skill-mismatch and onboarding delays are likely to have severely disadvantaged the participants involved in terms of their short-term career progression. Whilst the code school was useful for providing fundamental knowledge and skills, participants report that the majority of learning is done on the job after the code school has finished. Consequently, participants who performed well on the program all spoke of how important having a supportive team around them was when coming out of the code school was and how it helped them to succeed. In contrast, many of the participants that drew the short straw during the onboarding process and did not end up in a suitable team either left the firm as a result - deciding to apply their newly acquire coding skills elsewhere - or reverted back to a non-technical role within the firm.

Based on this, it seems likely that: if you were to take two identical program graduates that were equally capable and put them in two separate teams, one where the newcomer was supported in their development after coming out of the code school and the other without adequate support, each graduate would achieve vastly different outcomes after the program. It is therefore absolutely vital that these issues are ironed out in future iterations of reskilling programs of this type in order to prevent unlucky participants from being unfairly disadvantaged. The onboarding process needs to be made as consistent as possible across all program graduates so that equal opportunities can be achieved. Within large firms with a wide variety of team types and sizes, this is likely to be very difficult to achieve, but something that must be strived for nonetheless.

Older Participants

As illustrated by Figure 13 in the previous chapter, technologists graduating from the program had a higher proportion of older employees than externally hired software engineers, with X% of

program graduates being above the age of 45, compared to only Y% of external hires. It is encouraging to observe that the program is therefore providing opportunities for older employees to transition in to technology-based roles than would otherwise been available whilst at the same time increasing the diversity of the technology personnel within the firm. Furthermore, owing to their long careers and ample experience, older employees are likely to be able to contribute more business knowledge which, as already mentioned, is highly valuable within Agile teams.

However, qualitative interviews with program participants revealed that a considerable number of older employees struggled to keep up with the pace of the code school and took a longer time to grasp the technical concepts being taught than their younger coworkers:

> *"There were younger people who were really familiar with the more technical stuff, and then we had some of the old folks who had been with the company for 20+ years doing the same type of thing, so learning something new and fast was a real struggle for them."*

In future reskilling programs, one potential remedy for this issue would be to provide ample pre-work for students before they attend the code school, to ensure that older students have an adequate amount of time and materials to prepare themselves for the course so that they can hit the ground running when it begins. Owing to the diversity and non-technical knowledge that older employees can provide within Agile development teams, it is important that they are not shunned in favor of younger candidates that are likely to pick up the skills quicker (Park *et al.*, 2002). That is to say, the additional value provided by older employees is likely to be worth the additional time and effort required to train them.

Subsequent Reskilling Programs

Since the launch of this program, the firm has decided to apply the lessons learned from its implementation to design smaller, more specialized, reskilling programs that aim to fill specific skill gaps within the company. In late 2018, not long after the completion of the reskilling program, the firm launched a new program designed to teach Systems Administrators - who are typically responsible for the configuration and upkeep of the firm's computer systems and servers - how to write scripts that allow them to automate and streamline many of their tasks. Much of a system administrator's job had previously involved repetitive tasks (typically: logging in to a computer, opening up a command prompt, typing in some code, executing it, logging off and moving on to the next computer to repeat the process). Now, instead of having to manually log in to each individual computer, they are taught to write a program that is able to log in to hundreds of computers and perform these tasks for them in a fraction of the time.

This newly introduced program can be thought of as upskilling rather than reskilling, as the system administrators who take part remain in their current roles upon completion of the program, only they now have an enhanced skillset that allows them to perform their tasks more efficiently. One of the key differences between this program and the original program described in this paper is that it does not require the participants to leave the current jobs to attend a code

school. Instead, they are allotted a specified amount of time during their workday to complete online courses designed to provide them with the newly required skills. Allowing for the employee to remain working within their teams whilst they undergo their training results in significantly less disruption caused to the life of the participant – both professionally and personally – as well as to their team members and the firm as a whole.

The failure of the data science cohort back in late 2016 underlined the challenge of training employees with limited prior technical experience to become industry-level data scientists within the space of three months. In recognition of this, the firm is now considering running another data science reskilling program but this time raising the entry requirements so that only employees who are currently employed within the firm as software engineers are eligible to apply. It is hoped that through already having a lot of the requisite skills for being a data scientist, such as problem-solving capabilities and the ability to code using programming languages, software engineers will fare better than employees with limited prior technical knowledge.

To summarize, the main impact that the program had on the firm was the retention of business knowledge through reskilling business professionals to work within software development teams, thus bridging a gap between the technology and business arms of the company. This chapter has also discussed numerous ways in which the reskilling program has impacted employees, for better or for worse. This includes increased job retention for many RIF-eligible employees who were at-risk of losing their place within the firm had it not been for the program, and inefficiencies in the onboarding process that unfairly disadvantaged several unfortunate code school graduates. Overall, this chapter provides several invaluable lessons for future reskilling programs.

Chapter 7: Conclusions and Future Work

It would be fair to say that this particular exercise in employer-provided reskilling has been an overall success for the firm. Perhaps most crucially, they were able to retain valuable firm- and industry- related knowledge and bridge the gap between the IT and business arms of the company. In addition, they were able to reskill some of their existing workers rather than let them go and try to hire new workers, particularly during a very tight labor market. Participants coming from roles such as business systems analysts were able to combine their prior experience and highly-developed "people skills", such as the ability to communicate effectively and work within a team, with newly-acquired coding and technical abilities in order to become a highly valuable member of a an agile technical team.

The low turnover rate of program graduates when compared with that of externally hired software engineers is another positive outcome from the firm's point of view. Figure 23 shows that 90.1% of program graduates remained employed at the firm 18 months after beginning the program, whereas only 79.8% of externally hired software engineers remained employed 18 months following the date they were hired. The firm also stands to benefit from adapting the program in order to target and reskill specific groups of employees to provide specialized training. This is already being done with System Administrators, who are being taught to write software scripts to automate many of their tasks and is likely to also occur with future iterations of the program aimed at teaching data science to software engineers.

As one might expect, the firm's strategic decision to reskill its workforce also resulted in several positive outcomes for employees that took part in the program. Firstly, 131 RIF eligible employees that were previously at-risk of being let go by the firm ended up being successfully reskilled and found new roles within the company. Furthermore, the program came as a golden opportunity for employees that wanted to move into technical roles but had previously lacked either the time or financial means to reskill themselves. Finally, the skills that were learnt on program provided upward mobility for many younger and less experienced participants, opening up a range of new career paths and allowing them to negotiate higher salaries.

However, there were some individuals who drew the short straw during the onboarding process. Some participants were left without a team for up to a year, resulting in considerable stress and uncertainty along with skill atrophy as a result of having insufficient opportunities to put their new technical knowledge and skills into practice. Others were placed on teams and assigned roles that did not match what they had been trained for at code school, thus making it difficult for them to prove that they could add value in their role during the nine-month assessment period. These incidents of long delays and skills mismatch between participants and their eventual teams during the onboarding process reflect a lack of planning on part of the firm. Due to the importance of novice software engineers being placed in a supportive team environment in order to properly nurture and develop their technical skills, it is vital that firms ensure a smooth and consistent

onboarding process when implementing technical reskilling programs similar to the one studied here.

Analysis showing that RIF-eligible employees had worse outcomes from the program, both in terms of post-program job grade and likelihood of remaining in a program-related role, compared to non-RIF-eligible employees is another issue that needs to be taken in to account when designing future reskilling programs. If some level of trade-off exists, as this study would suggest, between selecting participants based on whether their position at the firm is under threat or based on their likelihood to succeed in a technical role, then the fundamental aims of the reskilling endeavor needs to be firmly established from the outset. That is, the firm must decide if their core aim is to keep the number of terminations relating to skill obsolescence as low as possible or to find the candidates within the organization that are most likely be most effective at filling skill gaps, and then design the program accordingly.

By taking part in the program, participants were required to leave their old roles at the firm, thus taking a considerable gamble regarding their future career prospects. With 42% of program participants having no prior experience in coding, it should be expected that at least some will end up deciding that a technology role is not right for them midway through the program. Many participants would therefore have benefited from the firm making an option to exit the program available for employees that turned out to not be a good fit.

Our findings suggest that firms designing similar reskilling programs as the one implemented in this case study should remain mindful that older workers are likely to take slightly longer to develop technical skills, such as coding, than their younger coworkers. Despite this, older and longer tenured employees are likely to bring more to the table in terms of aforementioned business- and industry-related knowledge and are therefore worth investing the additional time and effort required in order to be reskilled to the same level as younger employees.

Our analysis finds that the single biggest indicator of a participant's success on the program is whether or not they have any prior coding experience. This is evidenced by the fact that participants with prior coding experience listed on their resume were assigned post-program job grades on average 0.4 points higher than those without any prior coding experience and were almost twice as likely be in a program-related role nine months after completing the program. This finding suggests that, whilst formal reskilling programs have been shown to be effective in allowing workers to move in to coding-based roles, individuals should be proactive in seeking out opportunities to learn how to code early on in their careers - either through formal academic study or through more self-taught means - in order to be most adequately prepared for such reskilling opportunities when they do present themselves. From the firm's point of view, it would be worth exposing employees to coding opportunities in advance of future reskilling programs so that participants are able to hit the ground running and are not too overwhelmed by the course content. This also would allow prospective participants to have a better idea of whether or not coding is something that truly interests them. One approach to this could be to require all employees to complete an online coding course, such as those offered by Pluralsight, Coursera or EdX, before applying to the program.

Finally, demographic analysis found that the program provided more opportunities for females and older employees to obtain technical roles, when compared with hiring externally. Although females had a slightly lower acceptance rate than male applicants (18% compared with 22%), there was a higher proportion of female employees within the pool of program graduates (37.7%) compared to externally hired software engineers (13.4%). However, only 13.4% of minorities applying to the program were accepted, compared to 22.6% of non-minorities. The fact that minorities constitute 36.0% of all externally hired software engineers but only 16.3% of program participants further suggests that the program should have been more inclusive in terms of racial diversity.

Perhaps the greatest limitation of this study was the strong likelihood of selection bias present in the sample of program participants and technology managers that were interviewed during the qualitative analysis phase. As we relied upon the firm to provide us with the names and contact details of all interviewees, there is a possibility that they may have chosen program participants that performed particularly well on the program and therefore had particularly favorable opinions of it. Thus, it is worth taking into account when interpreting these results that their opinions may not have been representative of the population of participants as a whole.

Future Work

One of the aims of this study was to provide a broad overview of the reskilling program as well an insight into the range of different skills and skill gaps present within the firm, with the hope that this will help to guide any future avenues of research related to this particular case study. Beyond examining the effect that this program had on employees that chose to reskill, a follow-up study could involve an investigation in to different the outcomes of RIF-eligible employees who either did not apply to the program or were rejected. Those that remained at the firm were either able to hold on to their original jobs or managed to find new roles. The Sankey diagram in Figure 25, combined with the data presented in Table 2, show that 65% of RIF-eligible employees who applied for different roles at the firm outside of the program were still employed as of 12/1/2019. It is likely that many of these individuals were successful in applying for the new Agile-specific roles such as Product Owner and Scrum Master. Thus, further study into what it took for these employees to remain at the firm and how they were able to adapt to the agile transformation – through adopting new skills or utilizing past experience in order to find a new niche within the firm – would certainly be of great value from a skills resiliency viewpoint.

As discussed in the literature review, past attempts at reskilling workers have predominantly been through government-funded reskilling programs (Jacobson, 2005; Callan and Bowman, 2015), with a noticeable lack of documented firm-led reskilling exercises within the academic literature. However, the overall positive outcomes for both the firm and the worker observed in this study suggest that firms may be ideally placed to play a central role in reskilling the workers of tomorrow. Therefore, from a policy standpoint, future research could focus on comparing the outcomes of publicly- and privately-run reskilling programs. Furthermore, it would also be interesting to assess whether governments should consider partnering with employers in

both the public and private sector in order to achieve mutually beneficial outcomes from the perspective of firms, workers, and society as a whole. This could involve, for example, governments providing incentives such as subsidies to support firm-led reskilling programs of this kind.

References

Acemoglu, D. and Pischke, J.S., 1999. Beyond Becker: Training in imperfect labour markets. The economic journal, 109(453), pp.112-142.

Aguinis, H. and Kraiger, K., 2009. Benefits of training and development for individuals and teams, organizations, and society. Annual review of psychology, 60, pp.451-474.

Amazon Inc., (2020). *Upskilling 2025*. Accessed 28 May 2020, https://www.aboutamazon.com/working-at-amazon/upskilling-2025

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R. and Kern, J., 2001. Manifesto for agile software development.

Becker, G.S., 1962. Investment in human capital: A theoretical analysis. Journal of political economy, 70(5, Part 2), pp.9-49.

Bidwell, M., 2011. Paying more to get less: The effects of external hiring versus internal mobility. Administrative Science Quarterly, 56(3), pp.369-407.

Bidwell, M., Briscoe, F., Fernandez-Mateo, I. and Sterling, A., 2013. The employment relationship and inequality: How and why changes in employment practices are reshaping rewards in organizations. Academy of Management Annals, 7(1), pp.61-121.

Callan, V. and Bowman, K., 2015. Industry restructuring and job loss: helping older workers get back into employment. National Centre for Vocational Education Research.

Corbett, C. and Hill, C., 2015. Solving the Equation: The Variables for Women's Success in Engineering and Computing. American Association of University Women. 1111 Sixteenth Street NW, Washington, DC 20036.

De Grip, A., Fouarge, D., Montizaan, R. and Schreurs, B., 2019. Train to retain: Training opportunities, positive reciprocity, and expected retirement age. Journal of Vocational Behavior, p.103332.

De Grip, A. and Van Loo, J., 2002. The economics of skills obsolescence: a review (pp. 1-26). Elsevier Science.

Dostie, B., 2015. Who benefits from firm-sponsored training?. IZA World of Labor.

Dymock, D., Billett, S., Klieve, H., Johnson, G.C. and Martin, G., 2012. Mature age ‘white collar’workers’ training and employability. International Journal of Lifelong Education, 31(2), pp.171-186.

Field, J. and Canning, R., 2014. Lifelong learning and employers: Reskilling older workers. In International handbook on ageing and public policy. Edward Elgar Publishing.

Haelermans, C. and Borghans, L., 2012. Wage effects of on-the-job training: A meta-analysis. British Journal of Industrial Relations, 50(3), pp.502-528.

Heckman, J.J., Lochner, L.J. and Todd, P.E., 2003. Fifty years of Mincer earnings regressions (No. w9732). National Bureau of Economic Research.

Hey, T. ed., 2009. The fourth paradigm: data-intensive scientific discovery (Vol. 1). Redmond, WA: Microsoft research.

Jacobson, L., LaLonde, R.J. and Sullivan, D., 2005. The impact of community college retraining on older displaced workers: Should we teach old dogs new tricks?. ILR Review, 58(3), pp.398-415.

Kukhnavets, P. (2018). Why Agile is So Popular in Project Management World. Hygger.io, [online]. Available at: https://hygger.io/blog/why-agile-is-so-popular-in-project-management/#:~:text=It%20gives%20teams%20an%20opportunity,features%20and%20work%20in%20general. [Accessed 22 May. 2020.

Manyika, J., Lund, S., Chui, M., Bughin, J., Woetzel, J., Batra, P., Ko, R. and Sanghvi, S., 2017. Jobs lost, jobs gained: Workforce transitions in a time of automation. McKinsey Global Institute, 150.

McCracken, H. (2019). How Disney is turning women from across the company into coders. Fast Company, [online]. Available at: https://www.fastcompany.com/40576156/most-creative-people-2018-nikki-katz-disneycom [Accessed 22 May. 2020.

Mohrenweiser, J., Zwick, T. and Backes-Gellner, U., 2019. Poaching and firm-sponsored training. British Journal of Industrial Relations, 57(1), pp.143-181.

Moogk, D.R., 2012. Minimum viable product and the importance of experimentation in technology startups. Technology Innovation Management Review, 2(3).

Park, D.C., Lautenschlager, G., Hedden, T., Davidson, N.S., Smith, A.D. and Smith, P.K., 2002. Models of visuospatial and verbal memory across the adult life span. Psychology and aging, 17(2), p.299.

Royce, W.W., 1987, March. Managing the development of large software systems: concepts and techniques. In Proceedings of the 9th international conference on Software Engineering (pp. 328-338).

Sauermann, J., 2015. Worker reciprocity and the returns to training: evidence from a field experiment.

Tharenou, P., Saks, A.M. and Moore, C., 2007. A review and critique of research on training and organizational-level outcomes. Human Resource Management Review, 17(3), pp.251-273.

TIOBE programming community index, 2020. Accessed 20 May 2020, < https://www.tiobe.com/tiobe-index/>.

Weber, L., 2019. Why companies are failing at reskilling. Wall Street Journal.