Age(ing) in software development

Park, G.W.A.

Award date:
2020

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Software Engineering and Technology Group

Age(ing) in software development

Master Thesis

G.W.A Park

Supervisors:
Dr. A. Serebrenik
Dr. rer. nat. S. Baltes
Dr. E. Constantinou
Dr. G.H.L. Fletcher

Version 1.0

Eindhoven, May 2020

Abstract

As the population continues to age [48], progressively more countries across the world adopt policies aimed at keeping older workers in the workforce. This also has the implication that there is, and will continue to be, an increasing amount of older software developers. However, various lawsuits against alleged ageism in software development companies and statements such as Mark Zuckerberg's (in)famous statement that "young people are just smarter" raises the question of how age and ageing is viewed in the context of software development.

We perform a mixed methods analysis of 20 online articles and blog posts we collect, using both qualitative and quantitative analysis methods. We use content analysis to online articles and blog posts as well as analysing forum posts on Hacker News related to the articles and blog posts. We further used card sorting to investigate employment strategies for older developers found in the articles and blog posts, and subcoding to investigate the age-related aspects found in the articles.

We find that age(ing) in software development is mainly presented as limiting, with people staying in software development being the main focus for the public discourse. Age-related aspects that are mentioned include social challenges, professional challenges, and career path aspects. We did not find a common consensus on whether age is actually limiting with reactions being split between agreeing, disagreeing, and purely reproducing other public discourse. The majority of the age-related aspects discussed in the public discourse are also be found in literature. However, some of these findings run counter to findings in the literature.

Preface

First of all, I would like to thank my supervisors Dr. Alexander Serebrenik and Dr. rer.-nat. Sebastian Baltes for their support guidance throughout my thesis, as well as for them proposing, and willingness in helping to execute the research for, this topic for my thesis. Their input and guidance helped keep me on track with the research, and their willingness to answer any questions I had, helped me gain a deeper understanding of performing such a research.

Secondly, I would like to thank Dr. Eleni Constantinou and Dr. George Fletcher for being part of my graduation committee, and taking time to evaluate and give feedback on my work.

I would also like to thank my family for their support throughout my thesis as well as throughout my study as a whole. I would also like to thank my friends, and special thanks to Ninke for supporting me even more than normal, who helped keep me from going mad and reminding me that there were other things to do besides working on this thesis.

George Park
Eindhoven, May 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the population continues to age [48], progressively more countries across the world adopt policies aimed at keeping older workers in the workforce. This also has the implication that there is, and will continue to be, an increasing amount of older software developers. For example, in the Netherlands, the number of software developers between 55 and 65 increased from 6000, 4.3% of all developers, in 2003 to 26000, 12% of all developers, in 2017 [38]. In the USA, the number of software developers of 55 to 64 years old increased from 87,000, 8.3% of all developers, in 2011 [45], to 175,000, 11% of all developers, in 2017 [46].

With the trend of an increasing amount of older software developers one could also reasonably expect companies to be open to more older developers and adjusting to accommodate an increased amount of older employees, especially with the increased focus on diversity and inclusion in the past years. However, there have been, and currently are, lawsuits against alleged age discrimination in tech companies. [1] [2] These lawsuits could indicate that software development is not a welcoming area for older developers, who might face age discrimination and that the increased focus on diversity and inclusion has been missing out on a group of people.

The increasing amount of older software developers, the lawsuits above, and other examples of younger people being preferred in the technology field, including software development, such as Mark Zuckerberg's (in)famous statement that "young people are just smarter" raises the question of how age and ageing is viewed in the context of software development.

Previous research on older developers in software development has mainly focused on the effects ageing has on the ability of older developers to continue learning and in performing tasks, such as programming, in software development. There are also studies that investigated how older software developers are perceived by other developers and how they perceive themselves. This leaves the question of how older software developers are perceived by the broader public.

It is important to perform research in this area as it can help us understand what how older developers are perceived and the challenges they are perceived to face. These perceptions and perceived challenges could be preventing older developers from working as well as they otherwise would be, or even being able to work at all. An example of this can be found in a study performed by Libby Brooke [6], where 71 employees of 10 small and medium-sized Australian IT firms were interviewed, Brooke found that developers in their 50's and 60's were experiencing pressure to retire. Understanding these perceptions and challenges can lead to methods being developed that allow older developers to increase their effectiveness. It can also help us understand how ageing is perceived in software development, which could, as for older developers, present developers with challenges which could be preventing them from working as well as they otherwise would be.

The way older developers are perceived could also be forcing them towards career paths outside of software development. The perception of ageing in software development could also be

[1] https://www.theguardian.com/technology/2019/jul/22/google-pays-11m-to-jobseekers-who-alleged-age-discrimination

[2] https://www.forbes.com/sites/janicegassam/2019/06/26/wework-sued-for-age-discrimination-learning-lessons-for-companies

discouraging people from joining software development, reducing the number of available software developers. Furthermore, age discrimination is illegal in many countries, and as shown with the lawsuits presented above, can result in companies facing legal consequences. In this thesis, we will focus on understanding the way older software developers are perceived with the challenges that older software developers face being left outside of the scope.

1.1 Problem Formulation

In a literature study we performed on scientific literature before this thesis, we investigated the effects of ageing for software developers in the work environment. We found that older developers are motivated by different factors, do not perform worse when compared to younger developers, and are perceived to be different from younger developers by developers.

Some aspects that previous research did not investigate are how older developers are seen by the general public, and how age and ageing is seen in relation to software development. This should also be investigated as language, and thus the public discourse, is a powerful tool and its influence on individuals should not be underestimated [37] [36]. Furthermore, the public discourse on age and ageing in software development seems to point at it being possible to be, or concern towards being, too old for software development [3] [4].

In this thesis we aim to investigate what is being said about age(ing) in software development and how the information about age(ing) in software development is being used and spread. We will investigate this in the context of the broader public rather than just the findings of developers as not only developers themselves contribute to the public discourse. This leads to the question of "Whose perception of age(ing) in software development are we interested in?". We have chosen to focus on the perception that writers have on this topic, which we will explain in more detail below.

We use writers as an umbrella term to include authors of articles and blog posts in addition to people who ask questions and give answers to questions on online forums. We will investigate how writers present age and ageing in software development as they mostly have an external view on software engineering compared to software developers themselves, with the exception of writers that are also, or have been, software developers themselves. Furthermore, writers represent a group which can more easily reach a wider audience through the use of articles, blogs, and forums.

Now we have decided on whose perception of age(ing) in software development to focus on, we will narrow down which aspects of age(ing) in software development we will investigate. We are, in particular, interested in how the public discourse on age in software development is reproduced in online media, as online media is more easily spread to a large audience, and thus which thoughts and information from the public discourse are actually being reproduced in the online media. This can then be compared to the effects of ageing for software developers in the work environment found in the previous literature study. Another aspect we will investigate is, how the authors react to the public discourse in the media.

All of the above leads us to the following research questions:

RQ1 How is the public discourse on age in software development reproduced in popular online media?
RQ2 Which age-related aspects are reported in popular online media?
RQ3 What is the reaction to the public discourse in the popular online media?
RQ4 How does the public discourse on age(ing) in software development compare to results on the effects of age(ing) in software development found in literature?

By answering these research questions we contribute to the research on age in software development and aim to help motivate future research on this topic by finding open questions in

[3] https://www.quora.com/Is-30-years-old-too-old-to-learn-computer-programming

[4] https://www.quora.com/I-am-24-years-old-and-just-started-learning-coding-I-want-to-be-a-programmer-Am-I-too-late-in-the-game

the public discourse as well as common discrepancies between findings in scientific literature and the messages being spread in the public discourse. It can also serve as an indication of whether recommendations from previous research have been put into practice and and as an indication of where more research still needs to take place.

Besides contributing to and helping to motivate research, we will reiterate valid suggestions made in previous research on how to counter issues on age(ing) in software development found in the public discourse for use by employers, as well as critically analysing suggestions made in the public discourse.

1.2 Outline

In Chapter 2 we present a summary of a literature review performed previously on the effects of ageing for software developers, followed by the methodology of the research we performed in Chapter 3, after which we present the results of our analysis in Chapter 4. In Chapter 5 we will discuss the results found in our research and give a comparison to results found in the previous literature review. In Chapter 6 we present limitations and threats to the validity of our research. Lastly, in Chapter 7 we present the conclusions of our research and suggest areas for future research.

Chapter 2

Findings from Literature

In this chapter, we summarise the results found in a literature review performed in a seminar before this thesis, which can be found in its entirety in Appendix A. The research question we investigated in this literature review was "What are the effects of ageing for software developers in the work environment".

We will first shortly present the methodology used in the literature review, then we give an overview of the results found.

2.1 Methodology

In this section, we will shortly present the methodology used in the literature review performed previously. For a complete overview, see Chapter 2 in the literature review attached in Appendix A.

For the literature review, the snowballing procedure as given by Wohlin [47] was used.

Peer reviewed literature published before 2019, as well as non peer reviewed literature in the form of books or chapters from books published before 2019, on the following topics were included:

- Age stereotypes in IT
- Perception of older software developers
- Effects of ageing on software developers

Two search strings were used for the initial search, namely "perception ageing software developers" and "perception ageism software development". We chose to include "perception" in both search strings as we also wanted to find results on how software developers perceive ageing. We included "perception ageism software development" to find results on common ageist perceptions that are present in software development.

Google Scholar was used to search for a starting set of literature using the search string "perception ageing software developers" as well as the string "perception ageism software development". In total, 17,100 hits were found for the first search string and 7.820 hits were found for the second search string when not including patents and citations. Due to the limited amount of time available to conduct the research, the first 100 results of each search string were taken, as the relevance of the search results were found to drop off after the first 100 results. The union of these results was taken, giving a total of 200 results, on which the inclusion and exclusion criteria were applied, with ten candidates remaining that passed the inclusion and exclusion criteria. Using snowballing on these ten candidates, we then proceeded to add more relevant literature eventually finding a total of 18 relevant results. A full list of the literature found and included in the literature study can be found in Chapter 2 in the literature review attached in Appendix A.

2.2 Findings

In this section, we will shortly present relevant results from the literature review performed previously. For a complete overview, see Chapter 3, Chapter 4, and Chapter 5 in the literature review attached in Appendix A.

2.2.1 Ageing in general

We first looked at the effects of ageing on expertise as well as at findings in the literature on age stereotypes in the workplace. To do this, we used a review of age-comparative studies in different domains in their study on ageing and expertise by Ralf Krampe and Neil Charness [26], and a literature review on age stereotypes in the workplace by Posthuma and Campion [40].

Krampe and Charness found that although the results from psychometric test on general cognitive abilities may become worse as experts age, this does not mean that their performance in their field of expertise will also become worse. In addition, they found that high levels of skill can be maintained by older adults through deliberate practice.

Posthuma and Campion found stereotypes that older workers have a decreased ability to learn, have less ability, and are less motivated and productive than younger workers. Furthermore, they noted that these stereotypes are stronger in certain industries, including the IT industry. Not all age stereotypes found were negative though, as older workers were found to be "more stable, dependable, honest, trustworthy, loyal, committed to the job, and less likely to miss work or turnover quickly" [40]. They also found that work performance did not decrease with age and that individual skill and health are more important to work performance.

2.2.2 Ageing in Software Development

We next looked at effects of ageing for software developers in the work environment found in the 18 pieces of literature found during snowballing, including recommendations that have been made to combat the age stereotypes or challenges faced by older developers.

We first looked at the motivation of older software developers. In the context of free/open source software we found that older developers are motivated by different aspects, intrinsic motivation, community identification, when compared to younger developers, career benefits and learning, [10] [11].

In the context of open source, it was found that programming knowledge can be maintained at a high level into older age. It was also noted that older developers learn new technologies, and that knowledge of a technology can improve the knowledge of the successor of that technology in certain cases, which could aid older developers learning new technologies. These findings are presented in more detail by Morrison and Murphy-Hill [34], Morrison, Pandita, Murphy-Hill, and McLaughlin [35], and Kowalik and Nielek [25].

In the literature that was reviewed, we found that developers were found to think that being able to keep up to date and to be able to continue learning was linked to youth, and older developers would thus be less adaptive to change. Furthermore, we found that the demands placed on developers, such as working overtime and continuous learning, come into conflict with responsibilities, such as family needs, later on in life [6] [9].

In the career path of developers, moving to a non programming role was seen as a fundamental part of ones career, whereas expressing a wish to continue as a developer was seen as having a lack of drive [9]. Older developers were also seen as more expensive, not only due to higher pay, but also due to medical support [32] [49]. Even though the literature noted that older developers do not perform worse than younger developers and are seen as more expensive, it was found that employees aged 40 or over make only 88% of their younger counterparts, when controlling for education and technical experience levels [41].

It was also found that the IT industry may disadvantage older workers at the organisational level through the age differentiation of roles. Human resource management should investigate and put into effect measures to tackle these age stereotypes through the composition of teams as

well as influencing the age image and corporate culture through communications [9] [24]. It was also suggested that foresight planning that links industry and labour-market opportunities, and skills development should be provided to help software developers with their career. It could also be provided though part time work, flexible hours, contracting back to firms, and home based 'E-work' [6] [9].

2.2.3 Discussion

We will lastly present elements from the discussion of the results found in the literature review.

We found that there was a lack of a common definition of "older", with the term "veteran" also being used in some cases, in the literature. Some definitions include 50 years old and older to define "older", with the definition of "veteran" being at least 15 years of experience in software development and who are at least 40 years of age.

We found that the literature studied in this review provided certain recommendations as to how the current situation could be improved. As many of the papers in which these recommendations have been found, Brooke [6], Comeau and Kemp [9], Kleefeld [24], and Lünstroth [30], were published in or earlier than 2009, it could be investigated whether these recommendations have been adopted, and if they had an effect. If these recommendations have not been adopted, then they could be recommended again, possibly after being adapted to fit the current situation.

Chapter 3

Methodology

In this chapter, we discuss the methodology used in this research, the methodology of data collection as well as the methodology of the analysis performed on the data. As we have chosen to focus on the perception of age(ing) in software development in the online public discourse, we will limit the media we analyse in this thesis to online articles, online blog posts, and online forum posts. In this thesis, we perform a mixed methods research, which is the mixing of both qualitative and quantitative data collection and analysis methods, as described by Easterbrook, et al [13]. In particular, we perform what Easterbrook et al [13] refer to as a sequential exploratory strategy which begins with the collection and analysis of qualitative data, followed by the collection and analysis of quantitative data to assist with the interpretation of the qualitative results. To perform the analysis of the qualitative data, we conducted a content analysis [8] [27], which provides a systematic process for the analysis of selected concepts in qualitative data.

We will first briefly discuss content analysis, the data collection we performed, the inclusion and exclusion criteria for our data, the coding performed on the collected articles, the further analysis of coding results and analysis of employability strategies found in the data, and lastly, the collection of similar forum posts and analysis of the forum posts.

3.1 Content Analysis

Content analysis is a research method used to study and analyse texts, other means of communication or qualitative data [8] [27] [12]. It involves determining the presence of certain words or concepts in the texts, which are then used to draw conclusions about the messages within the texts and the context of the texts, such as the author(s) and the audience of the texts. Furthermore, content analysis is an unobtrusive research method, as it uses texts that already exist rather than the collection of data through surveys or other experimentation.

Content analysis is used in many different fields and for many different purposes such as the following examples, taken from Berelson [3]:

- To describe trends in communication content
- To identify the intentions and other characteristics of the communicators
- To reveal the focus of attention

An example of content analysis being used to examine communication towards software developers is Surakka [44], who investigates the most common technical skills sought in job advertisements in the USA.

A content analysis can be an inductive or deductive process [12]. An inductive content analysis is an exploratory research of a phenomenon, of multiple phenomena, with no previous knowledge. As such, the resulting concepts will be derived from the data. A deductive analysis on the other hand is used to investigate a hypothesis, and the categories for the data will be derived from previous knowledge. We will be performing a content analysis using an inductive process as we are not building upon an existing base of research, or working from an existing hypothesis.

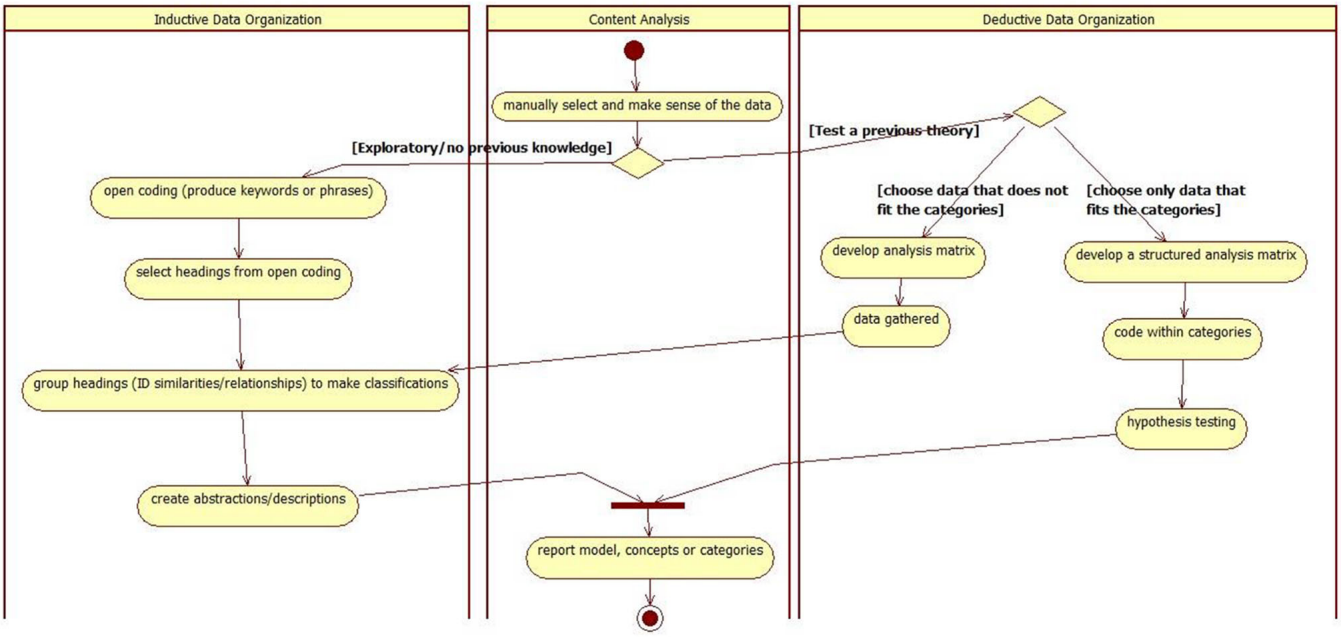


Figure 3.1: Content analysis UML activity diagram [12]

DeFranco and Laplante [12] present a process for performing a content analysis in software engineering in Figure 3.1. We will be following process on the left, called inductive data organization, as we are performing an inductive content analysis.

There are two main types of content analysis, conceptual analysis and relational analysis. Conceptual analysis focuses on the existence and prevalence of certain selected concepts, either words or more abstract concepts, in the data. Relational analysis does not just examine the existence and prevalence of selected concepts, but also examines the relations between the selected concepts in the data.

Of particular interest to us is that content analysis can be used to identify the focus or communication trends of a group, which is what we aim to do for the perception of age(ing) in software development in the public discourse. We will focus on a conceptual content analysis rather than a relational analysis as we focus on the existence and prevalence of selected concepts, such as age-related aspects reported in the online media.

3.2 Data Collection

3.2.1 Articles and blog posts

We use the Google Custom Search API [1] to find the articles and blog posts in the public discourse that we analyse in this research. We use the Google Custom Search API as it gave us more control over the querying process through the availability of additional query parameters compared with a normal Google search. We also make use of the Google Custom Search API as we do not want to limit our search to only sources of media known by the researchers. Furthermore, Google is where we expect our target audience, the general population, to search and find information on age(ing) in software development.

We use the following query in the Google Custom Search API:

$$q = \text{age software developer}, gl = \text{us}, lr = \text{lang\_en}$$

[1] https://github.com/sbaltes/api-retriever#example-5-retrieve-top-100-results-for-google-search-queries

For the query string, given by the parameter q, we choose to use *age software developer*. We include *software developer* because we are interested in what's being said about software developers rather than software itself, and *age* as we do not only want to focus on older or younger developers, but want to know what is being said about ageing in general. Furthermore, as we have decided to focus on discourse from the USA, we set the parameter gl, which emulates the geolocation of a user making the query by boosting search results whose country of origin matches the parameter value, to us for the USA. Finally, as we only look at discourse that is written in English, we set the parameter lr, which restricts the search to documents written in a particular language, to $lang\_en$.

We collect the top 100 search results using the settings described above.

After collecting the media using the Google Custom Search API on August 2, 2019, we gather the data that we require from the media. To analyse the articles and blog posts, we at least need to collect the text. We also store the link to the article or blog post to be able to refer back to it if necessary. We gather the definitions of old used in the articles and blog posts, as well as the publication date, for comparison against the definitions in literature, and publishing date of the literature. Lastly, we collect information on the author and publisher of the articles and blog posts as there may be differences in the reaction to the public discourse depending on the author's job and publisher type, blog post or article. For the articles and blog posts we thus gather the following information:

- The text of the article or blog post
- Link to the article or blog post
- How old was defined in the article
- Publication date
- Author information
  - Name of the author
  - Author's job
  - Location of the author
- Publisher information
  - Name of the publisher the article of blog post is published in
  - Publisher type (Blog/News publisher/etc.)
  - Country the publisher is based in

3.2.2 Forum posts

For the collection of the forum posts, we only look at Hacker News [2], a site that focuses on computer science and entrepreneurship on which users can post and discuss other news articles.

As mentioned above, we use what Easterbrook et al [13] refer to as a sequential exploratory strategy which begins with the collection and analysis of qualitative data, followed by the collection and analysis of quantitative data. This is done to assist with the interpretation of the qualitative results. Another option would be to use what Easterbrook et al [13] refer to as concurrent triangulation strategy, in which different methodologies, or different data sources, are used concurrently to cross-validate or confirm findings. As it can be the case that 'what people do' and 'what people say' can be different, analysing data from multiple data sources can help improve the validity of the research.

We use a sequential exploratory strategy rather than concurrent triangulation as, rather than analysing them concurrently to form hypotheses from the data as a whole, we want to use the forum posts to support or counter the hypotheses formed from the analysis of the articles and blog posts. We thus only include posts on Hacker News that discuss an article we include in our study. We do this by searching for the titles of the articles on Hacker News.

[2] `https://news.ycombinator.com`

3.3 Inclusion and Exclusion Criteria

After collecting the articles and blog posts, we apply inclusion and exclusion criteria to select relevant articles and blog posts that we will analyse. For our research questions, we investigate the public discourse and in particular, popular online media. Furthermore, we focus on the perception of age and ageing by writers, authors of articles and blog posts. As such, we only include written media, specifically online articles and blog posts. We excluded online forum posts, (job) advertisements, press releases, and websites that just reproduce statistical data as we are interested in original content. We do not place any limitations on the time period that the online articles, and online blog posts had to have been published on.

We include online articles, and online blog posts that deal, at least partially, with the following topic: "Perception of age or ageing for software developers".

Initially a single researcher evaluates whether an item collected in the data collection should be included or excluded, after which and before the coding of included items, the remaining two researchers go over the list of included and excluded items to check if the initial researcher did wrongly include or exclude an item.

3.4 Coding

After we had selected a body of public discourse to study and had excluded the results that do not fit our inclusion criteria, the next step was to analyse the data. We have chosen to perform qualitative coding on the data as it is a way of condensing and mapping the larger text based data that we work with into data that can then be more easily related to our research questions. In other words, the goal of coding the articles was to help create a framework with which to answer the research questions by creating codes and assigning parts of our data to them and filtering out data from the articles that does not discuss age(ing) in software development, as well as to help with a more detailed analysis of the texts later on. This is also in line with content analysis, in which we will then further analyse the results from the coding.

We will possibly perform multiple iterations of coding as we perform inductive coding, as explained in Section 3.1, where we do not start with a set a codes that we use, but rather create the codes based on the data. We do not fix the amount of iterations of coding we perform before hand, but rather make the choice of performing another iteration of coding or not at the end of an iteration. We perform another iteration of coding if we are missing certain information in the coding that we need to answer the research questions, at which point we add another question that we code for. If it is the case that we change the codes we use, or the structure of the codes, we perform another iteration of coding with those codes. If none of the above hold, we stop performing iterations of coding. We use books by Saldaña (2013) [42] as well as Gibbs (2007) [18] and Menzies, Williams, and Zimmermann (2016) [33] as guides during the coding process, as well as using them to select coding methodologies for the research. Qualitative coding is a way of iteratively analysing qualitative data by coding and recoding data. A code in this context is most often a word or short phrase that assign a certain meaning or attribute to a portion of data.

There are also different methods in which qualitative coding can be performed depending on what the needs of the research are. By using different coding methodologies in combination with multiple iterations of coding, data can be analysed in differing degrees of detail and from different perspectives. As we perform an inductive content analysis, as explained in Section 3.1, we perform open coding as part of the process of content analysis as shown in Figure 3.1, which is a process in which codes are created that describe or classify the data or observed phenomena in the data. The coding is performed by three researchers, of which two are more experienced researchers.

3.4.1 First iteration of coding

For the first iteration of coding, we perform exploratory coding of the data to investigate the initial themes and categories we find, which we can then refine for later iterations of coding. We perform

descriptive coding of the data, which summarises the topic of a section of text in a word or a short phrase. We use descriptive coding as, according to Seldaña [42], it is an appropriate method for all qualitative studies, particularly for qualitative researchers learning how to code, and for studies with a wide variety of data forms. Furthermore, applying descriptive coding leads to a categorised inventory or index of the data that has been coded, which allows for further coding steps of the resulting data. As we do not know what common themes and categories exist in the data, we do not start with a codebook, a collection of the codes with their content descriptions and an example of the code on some data, but rather create one by coding the media and creating codes.

In this iteration coding, we code the following items in the articles and blog posts:

- Whose voice is reflected?
- What is the main message of the paragraph?
- What category does the paragraph fit in? (The code we assign to the paragraph)

The articles and blog posts are coded by three researchers, of which two are more experienced researchers, using the following workflow:

One researcher copies over the text from the articles/blog posts into individual spreadsheets for all the researchers. After which, we individually code a number of articles/blog posts, during which we create our own codes for the coded items, with possibly multiple codes being assigned to an item. We code the articles/blog posts per paragraph, using the notation of paragraph, text that starts on a new line, in the original articles/blog posts. It is important to note that we do not include titles, abstracts, headings, or captions in our coding as we found that they either just repeat information that is already present in the text, or are not relevant for coding.

We discuss the results of coding and create codes that we all agree upon and add those codes to our codebook. We then continue to code the articles/blog posts using the codebook where we once again create a new code if none of the codes in the codebook fit and repeat the process of discussing the results for the newly coded articles.

At the end of the iteration of coding, we discuss and check whether another iteration of coding is required.

3.4.2 Second iteration of coding

After performing in initial iteration of exploratory coding, we continue with a second iteration of coding. The initial step in this iteration of coding was the creation of a codebook based on the experience and knowledge gained in the previous iteration of coding. A code can be added to the codebook if during coding it appears as a more common theme or category and all researchers agree that the code should be added. The final codebook we used to code the articles can be found in Appendix B.

As in the case of the first iteration of coding, we perform descriptive coding of the articles and blog posts. However, during this iteration of coding, we split the coding up into different sub-questions. In this way, we have a more structured approach to coding the data. We code the following items in the articles and blog posts:

- What developer type(s) are mentioned?
- Is age perceived as limiting?
- What is the speaker's reaction to the public discourse?
- How does the author reproduce or comment on the public discourse?
- Which reported age-related aspects are mentioned?

In this iteration of the coding, the articles and blog posts are once again coded by three researchers, of which two are more experienced researchers, using the following workflow:

We reuse the method of the previous iteration of coding, where one researcher copies over the articles/blog posts into individual spreadsheets for all the researchers. After which, we individually code the articles/blog posts, using the codes from the created codebook. We code the articles/blog posts per paragraph, using the notation of paragraph, text that starts on a new line, in the original articles/blog posts. As in the previous iteration of coding, we do not include titles, abstracts, headings, or captions in our coding.

At the end of the iteration of coding, we discuss and check whether another iteration of coding is required.

3.5 Combination of coding results

If we decide that no more iterations of coding are required, we merge the results from our final iteration of coding. We use the merged results to draw conclusions on the research questions, with the previous iterations of coding contributing the final codes and structure of coding used in the final iteration of coding. We will first combine the results before analysing the results from coding as it can serve as a way to discover results that would have otherwise been missed by a single coder. In order to combine the results, some kind of reliability check should be performed to see if the coders agree and remain consistent should be performed. We look at different ways to combine the results from the individual coding of articles and check the reliability of coding.

First, we investigate quantitative measures to measure coders agreement and consistency, also called intercoder agreement or interpretive convergence. Saldaña [42] points to multiple statistic measures that can be used for intercoder agreement, out of which, we look at the Fleiss kappa [16] and Krippendorf alpha [21] measures in more detail, as these work for more than two coders which is the case we had in our coding.

Secondly, we investigate qualitative measures to ensure coder agreement and consistency. Saldaña [42] mentions the use of group discussion, coder adjudication, and group consensus as ways to ensure an agreement of the final coding of data. Moreover, Gibbs [18] also mentions several techniques to control and check coder reliability. These include the checking the agreement of individually coded texts by a independent researcher, checking for definitional drift, where codes might be assigned differently in material that is coded later, and simply checking each others work when working in a team.

We decide against using quantitative measures to measure the agreement of our individual coding as we find that discussing the different results often leads to an agreement among all coders, as well as new thoughts and ideas being discussed. We also find the quantitative measures to be of less use to us as we also allowed for multiple codes to be assigned of which only a subset of the codes would be assigned by all coders, which these measures do not have a clear solution for. As such, we use a qualitative approach to merge the coding results. We combine the results of coding through the use of majority vote, where we assigned the codes to a paragraph that the majority of coders had individually assigned to it. If there was no majority code for a paragraph, or it was the case that disagreeing codes such $True$ and $False$ are assigned to it by different coders, we discuss the paragraph with all coders and code the paragraph together.

3.6 Further analysis of coding results

After the main coding of the articles and blog posts, we perform another set of analyses on certain aspects of the coding results, to gather either a more detailed view of the data, or another view on the data from another perspective. Note that the analysis of the articles and blog posts from the perspective of employability strategies was originally performed as part of a separate paper and has also been included in this thesis. The two further analyses were performed concurrently due to time constraints.

3.6.1 Subcoding

Once we had completed the coding of the articles and blog posts, we performed subcoding the results of coding for "Which reported age-related aspects are mentioned?" to create a more detailed view of the age related aspects we found.

Subcoding is intended as a way of creating subcategories, hierarchies, and indexes in the coded data. It is thus appropriate when the analysis of the data requires more detail than the general codes provide, and as a way organising the data. The process of subcoding is where a secondary

code is assigned after a primary code has been assigned to piece of data, with the secondary code providing more detail than, or enriching, the primary code. As in the main iterations of coding the data discussed in Section 3.4, we perform open coding, which is a process in which codes are created that describe or classify the data or observed phenomena in the data.

The subcoding of the results of coding for "Which reported age-related aspects are mentioned?" is performed by a single researcher due to time restraints. It was also performed concurrently with the analysis of the articles and blog posts from the perspective of employability strategies and thus the remaining two researchers were not available to assist in the coding.

3.6.2 Employability strategies

We also look at the articles and blog posts from the perspective of employability. As such, two researchers code the articles for employability strategies for older developers, from both the perspective of individual developers as well as from the perspective of companies. Rather than coding individual paragraphs as previously performed and described in Section 3.4, card sorting, as described by Zimmermann [51], is used.

Card sorting is a method of identifying common themes in data and is often used when creating mental models and deriving taxonomies. Card sorting involves three different phases. The first phase, called the preparation phase, is the phase in which the cards are created that are to be sorted. The second phase, the execution phase, is the phase in which the cards are sorted into groups. During the card sorting we perform, it is possible for articles to be assigned to multiple categories. In phase three, the analysis phase, structure is given to the created groups, creating high level themes or hierarchies of groups. In our case, we create a mind map of the resulting categories and the resulting hierarchy.

There are two types of card sorting, open card sorting and closed card sorting, where open card sorts do not use predefined groups, and closed card sort use predefined groups. We performed an open card sort, as we do not know which categories we can expect to emerge in advance, but rather let the categories emerge from the data.

3.7 Forum Posts

As mentioned in Section 3.2, we will use the forum posts to attempt to support our hypotheses drawn from the coding of the articles by finding further supporting statements, rather than for triangulation of any possible hypotheses drawn from the coding of the articles. Due to the larger amount of comments and coded paragraphs from the articles and the limited time to perform the research, we automate the matching of possibly similar forum posts and paragraphs. To do this, we create a program in Python [3] that computes the pairwise similarity of paragraphs from articles and posts from Hacker News. We use a book by Manning [31] on information retrieval to help with the creation of this program.

3.7.1 Similarity program

We first give a rough outline of the program below:

- Collect forum posts from Hacker News
- Perform preprocessing and normalisation on forum posts and coded paragraphs
- Perform tokenisation of the forum posts and coded paragraphs
- Calculate similarity score between forum posts and coded paragraphs

First, we need to collect the posts from Hacker News. To do so, we use the Hacker News API [4], by finding the id's of the articles that we include, see Section 3.2, and traversing the tree of comments for each post and storing each comment.

[3] https://www.python.org/
[4] https://github.com/HackerNews/API

Secondly, we perform preprocessing and normalisation of the comments and the paragraphs from the articles as is discussed in Chapter 2.2 Manning [31]. We first convert all text to lowercase and remove punctuation as well as HTML from the Hacker News comments. After performing the simple preprocessing, we remove stop words, a set of commonly used words, from both the Hacker news posts as well as the paragraphs using the stop word list provided by NLTK [4] for Python. Next, we look into stemming or lemmatisation of the text. Stemming is a process that reduces words through a heuristic process, whereas lemmatisation makes use of a vocabulary and morphological analysis of words to try and reduce a word to its base, or dictionary for or lemma. Manning [31] mentions that stemming can increase recall while harming precision. As we are only looking for similar posts when compared to the paragraphs from the articles, we have thus chosen to use stemming rather than lemmatisation. Some commonly used stemmers are the Porter stemmer, the Snowball stemmer, and the Lancaster stemmer [5]. We use the snowball stemmer [4], which is an improved version of the Porter stemmer [6], to perform the stemming.

We next look into the tokenisation of the text. In particular, we look at two different options for the tokenisation of the text, namely, n-grams, and w-shingles. N-grams are contiguous sections of the text of n items, in this case characters. W-shingles, also called word n-grams, are contiguous sections of w items where the items are words in this case. We use literature in which these methods are discussed and used to investigate which method would fit our situation best and whether there are choices for n or w that generally provide the best results. We look at how n-grams or w-shingles were used in articles by Kanaris et al [23], who look at the use of n-grams and w-shingles for spam detection, Lopez-Gazpio et al [29], who look at the use of w-shingles for sentence similarity and inference, Gali et al [17], who discuss using n-grams to find text similarity is discussed, and Stefanovič et al [43], who use w-shingles to detect text similarity.

We can not decide whether to use n-gram or w-shingles and what size of n or w we should use based on literature itself. As such, we run our program with different settings for tokenisation that we find in the literature. We run our program with the following configurations for tokenisation:

- n-grams of size $n \mid n \in \{3, 4, 5\}$
- w-shingles of size $w \mid w \in \{1, 2, 3\}$

Once we gather the results, we decide on which settings to use which will be discussed in Chapter 4.

After preprocessing, normalisation, and tokenisation of the comments and the paragraphs from the articles, we compute the pairwise similarity of the comments and the paragraphs. We measure the pairwise similarity of the comments and the paragraphs using the term frequency, with the terms depending on which tokenisation method was used, and cosine similarity which is described in Chapter 6.3.1 Manning [31]. The standard cosine similarity equation is given in Equation 3.1 where A and B are vectors containing the term frequencies for the terms we compare. Rather than using the standard cosine similarity measure, we opt to use the BM15 weighting scheme, which intends to lower the effect of frequent terms in the comparison. Using the BM15 weighting scheme, we apply the BM15 weighting on the term frequencies in A and B resulting in vectors A^* and B^* with the term frequencies as shown in Equation 3.2 and Equation 3.3, resulting in the similarity equation given in Equation 3.4. In Chapter 11.4.3 in Manning [31], Manning talks about the choice k in Equation 3.4 and notes that experimentation has found that $1.2 \leq k \leq 2$ are reasonable values. As we do not have time to create a test set and evaluate the performance of different values of k, we choose to set the $k = 1.5$ in Equation 3.2 and Equation 3.3.

$$similarity(A, B) = \frac{AB}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{3.1}$$

$$A_i^* = \frac{(1+k)A_i}{A_i + k} \tag{3.2}$$

[5] https://stackoverflow.com/questions/10554052/what-are-the-major-differences-and-benefits-of-porter-and-lancaster-stemming-alg

[6] https://tartarus.org/martin/PorterStemmer/index.html

$$B_i^* = \frac{(1+k)A_i}{A_i + k} \tag{3.3}$$

$$BM15\ similarity(A^*, B^*) = \frac{\sum_{i=1}^{n} A_i^* B_i^*}{\sqrt{\sum_{i=1}^{n} A_i^*}\sqrt{\sum_{i=1}^{n} B_i^*}} \tag{3.4}$$

Combining the use of different tokenisation options discussed above with the use of BM15 weighting, we thus run our program with the following configurations:

- n-grams of size $n \mid n \in \{3, 4, 5\}$ with cosine similarity with the BM15 weighting scheme with $k = 1.5$
- w-shingles of size $w \mid w \in \{1, 2, 3\}$ with cosine similarity with the BM15 weighting scheme with $k = 1.5$

3.7.2 Selection of similar forum posts

The next step in analysing the forum posts, is the selection of the most similar forum posts compared to the individual paragraphs. Before we select a certain cutoff point, we test whether the distribution of the similarity scores between the forum posts and the paragraphs from the articles. If the data follows a normal distribution, we can use statistical rules to select similar forum posts above a certain cutoff point, else we will assign a certain number of the highest scoring forum posts to the paragraph.

We perform the Shapiro-Wilk test, and the Anderson-Darling test, as well as create QQ plots for the distributions, to test for normality for the distribution of the similarity scores. We use multiple tests to check for normality due to certain weaknesses possibly causing us to reject normality incorrectly when using a single test. If we can conclude that the distribution of the of forum posts is normal, we can use the fact that 95% of the similarity scores are within two standard deviations of the mean in a normal distribution. We can thus select scores that are greater $\mu + 2\sigma$, where μ is the mean of the similarity scores and σ is the standard deviation, to select the highest 2.5% scoring results.

In the case that the distribution of the similarity scores does not follow a normal distribution, we assign the five highest scoring comments to the selected paragraph. We ensure that all paragraphs have unique comments assigned to them by only assigning duplicate comments to the paragraph with which it has the highest similarity score, replacing the comment with the next highest scoring comment for the other paragraphs for which the comment occurs.

3.7.3 Analysis of forum posts

After assigning the most similar forum posts to coded paragraphs from the articles and blog posts, we proceed to analyse the selected forum posts. As mentioned in Section 3.2, we will use the forum posts to attempt to support our hypotheses drawn from the coding of the articles.

The first step in analysing the forum posts, is to check whether they are actually relevant or not, by checking whether the forum post and the paragraph have the same topic. If the forum post and the paragraph from the article or blog post talk about the same topic, the next step is to check if the forum post agrees or disagrees with the paragraph. The coding of the forum posts, whether they are relevant or not and whether they agree or disagree with the corresponding paragraph, is performed by two researchers. It is ensured that there is an overlap between the coded forum posts between the two researchers to check for disagreement. These disagreements are resolved by a third researcher.

Chapter 4

Results

In this chapter we present the results from the data collection, the coding of the articles, the further analysis performed on the coding results, and the collection of similar forum posts. The query results are available online https://bit.ly/2z19eXW together with our retrieval tool https://bit.ly/2XEuWez. The final merged coding results are available online at https://bit.ly/36b0MlG, with the online articles and blog posts that we included being available online here https://bit.ly/2Tg4uFl.

4.1 Data Collection

4.1.1 Articles and blog posts

We selected the top 20 relevant articles and blog posts that were in included in our data as further relevant results were becoming more sparse in the search results as well as limited time to perform the research. These 20 relevant results were present in the first 63 results gathered from the data collection as discussed in Section 3.2. A list of titles and links to the articles we have included in our research can be found below.

- **A1: Silicon Valley's obsession with youth, summed up in one chart** https://www.businessinsider.com/silicon-valley-age-programmer-2015-4
- **A2: Is there a software developer age limit? Apparently, it's 45** https://searchsoftwarequality.techtarget.com/opinion/Is-there-a-software-developer-age-limit-Apparently-its-45
- **A3: Software Developers Are Terrified Of What Happens When They Hit 30** https://www.businessinsider.com/software-developers-fear-age-30-2014-3?international=true&r=US&IR=T
- **A4: Where do all the old programmers go?** https://www.infoworld.com/article/2617093/it-careers-where-do-all-the-old-programmers-go.html
- **A5: How to retrain to be a software developer at 30 in an age-biased industry** https://medium.com/@SamOrgill/how-to-retrain-to-be-a-software-developer-at-30-in-an-age-biased-industry-570eccfd276b
- **A6: 35 isnt too old to work in tech but you may feel over the hill, say software engineers** https://www.cnbc.com/2017/09/18/even-35-year-olds-may-feel-ageism-in-tech-google-amazon-engineers.html
- **A7: No place for the old? Is software development a young person's game?** https://www.techrepublic.com/article/no-place-for-the-old-is-software-development-a-young-persons-game/
- **A8: Stories from 300 developers who got their first tech job in their 30s, 40s, and 50s** https://www.freecodecamp.org/news/stories-from-300-developers-who-got-their-first-tech-job-in-their-30s-40s-and-50s-64306eb6bb27/

- **A9: As A Software Developer, How Can I Ensure I Remain Employable After Age 50?** https://www.forbes.com/sites/quora/2012/09/17/as-a-software-developer-how-can-i-ensure-i-remain-employable-after-age-50/
- **A10: Older programmers are more knowledgeable, but harder to find** https://www.itworld.com/article/2710015/older-programmers-are-more-knowledgeable--but-harder-to-find.html
- **A11: Do programmers die at the age of 40?** http://netaq.ae/blog/do_programmers_die_at_the_age_of_40/
- **A12: The Tech Industrys Darkest Secret: Its All About Age** https://www.linkedin.com/pulse/20130422020049-8451-the-tech-industry-s-darkest-secret-it-s-all-about-age/
- **A13: Being a Junior Developer at 30** https://hackernoon.com/being-a-junior-developer-at-30-38309f1daee8
- **A14: The Old (Age 35) American Software Developer** https://normsaysno.wordpress.com/2019/04/01/the-old-age-35-american-software-developer/
- **A15: Programmers: Before you turn 40, get a plan B** https://improvingsoftware.com/2009/05/19/programmers-before-you-turn-40-get-a-plan-b/
- **A16: Can You Be Too Old For Software Development?** https://dzone.com/articles/can-you-be-too-old-software
- **A17: Silicon Valleys Dark Secret: Its All About Age** https://techcrunch.com/2010/08/28/silicon-valley%E2%80%99s-dark-secret-it%E2%80%99s-all-about-age/
- **A18: Ageism in tech: the not-so-invisible age limit developers face** https://bdtechtalks.com/2019/03/29/ageism-in-tech-age-limit-software-developers-face/
- **A19: Tech workers worry about age discrimination at age 40: Study** https://www.theladders.com/career-advice/older-tech-workers-fear-age-discrimination
- **A20: Returnships' Help Stay-at-Home Moms Get Back to Work** https://www.nbcnews.com/better/careers/returnships-help-stay-home-moms-get-back-work-n750651

4.1.2 Metadata articles and blogposts

In Table 4.1, we present the different definitions of "old" used in the articles and blog posts analysed and how often they occur in the 20 articles and blog posts coded. We find that 30 years old and older is the lowest definition of "old" used, with 45 years old and older being the highest definition of "old". The most commonly used definition that we found, is 40 years old and older. Moreover, the fear of being considered too old to be a software developer could start at an even younger age, as in article A8 we find a list of Quora posts of people, aged from 14 to 60, that are concerned about being too old to join software development.

Definition of "Old"	Count
30+	4
35+	3
35-40+	2
40+	6
45+	2
No explicit definition	3

Table 4.1: Definition of "Older" in the coded articles and blog posts

In Table 4.2, we present the publication year of the 20 articles and blog posts we coded. The articles and blog posts that we coded, date from between 2009 and 2019, with the most articles and blog posts being published in 2017 and half of the total articles and blog posts being published in 2017 or later.

Publication year	Articles and blog posts
2009	A15
2010	A16, A17
2012	A4, A9
2013	A10, A12
2014	A3
2015	A1,
2016	A7
2017	A5, A6, A11, A13, A19, A20
2018	A2, A8
2019	A14, A18

Table 4.2: Publication year of the coded articles and blog posts

4.1.3 Forum posts

For the forum posts on Hacker News, we found ten discussions on six out of the 20 articles and blog posts included on February 24, 2020. For one article we found two discussions, however, as the two discussions did not contain any comments, we did not include them. For the remaining 13 articles and blog posts, no discussions were found on Hacker News. The following posts on Hacker News were found and included:

Article A4: Where do all the old programmers go?
- https://news.ycombinator.com/item?id=12743697

Article A10: Older programmers are more knowledgeable, but harder to find
- https://news.ycombinator.com/item?id=5679321

Article A12: The Tech Industrys Darkest Secret: Its All About Age
- https://news.ycombinator.com/item?id=5600538

Article A15: Programmers: Before you turn 40, get a plan B
- https://news.ycombinator.com/item?id=20592384
- https://news.ycombinator.com/item?id=16934500
- https://news.ycombinator.com/item?id=9361580
- https://news.ycombinator.com/item?id=650437

Article A17: Silicon Valley's Dark Secret: It's all about Age
- https://news.ycombinator.com/item?id=1641763
- https://news.ycombinator.com/item?id=9710936

Article A19: Tech workers worry about age discrimination at age 40: Study
- https://news.ycombinator.com/item?id=15525814

4.2 First Iteration of Coding

We performed the first iteration of coding to investigate which initial themes and categories would emerge from the data, which we could then refine for later iterations of coding.

During the first iteration of coding, we coded the top ten articles and blog posts that were included in our data during this iteration of coding, resulting in a total of 171 paragraphs being coded. After the removal of paragraphs that were only assigned the code N/A, and were thus seen as not relevant by the majority of the coders, 115 coded paragraphs remained to be analysed.

The following codes emerged and were used in the first iteration of coding
- Positive Personal Experience Older Developers
- Negative Personal Experience Older Developers

- Social Challenges of Older Developers
- Professional Challenges of Older Developers
- Younger Developers are Preferred
- Continuous Learning/Continuous Change
- Age vs Experience
- Age vs Innovation
- Acknowledgement of Ageism
- Attributes of Older Developers
- Career Path Software Developers
- Use of authority figures
- Statistics
- Explanation of Age Distribution
- Career Choice Advice
- Public discourse about age in software development
- N/A

We found that the codes we had created during this iteration of coding were not sufficient to answer our research questions, with there being overlap between codes, making it more difficult to analyse. We found that the coding structure we were using, as can be found in Section 3.4.1, did not have sufficient depth in the amount and type of questions that we coded for. As such, we decided to perform another iteration of coding, where we would start by creating a codebook, using some of the usable codes created during this iteration of coding, before re-coding the articles themselves. We also adapted what we coded for the next iteration of coding, creating a new structure based on our experience from coding in this iteration, as can be found in Section 3.4.2.

4.3 Second Iteration of Coding

During the second iteration of coding, we coded the top 20 articles and blog posts that were included in our data during this iteration of coding, which includes re-coding the articles coded in the previous iteration of coding using the new codes and codebook, resulting in a total of 399 paragraphs being coded. After the removal of paragraphs that were only assigned the code N/A, and were thus seen as not relevant by the majority of the coders, 245 coded paragraphs remained to be analysed.

For this iteration of coding, we decided to first create a codebook, adapting codes and subject knowledge gained from the previous iteration of coding. As such, we created codes linked to each research question, excluding RQ4: "How does the public discourse on age(ing) in software development compare to results on the effects of age(ing) in software development found in literature?". The final codes for these questions will be discussed in more detail below, and our codebook with the final codes can be found in Appendix B.

We also coded the articles and blog posts for the questions "What developer type(s) are mentioned?", "What is the speaker's reaction to the public discourse?" and "Is age perceived as limiting?" to help answer the research questions in more detail. The final codes, a short description of the code for these questions we coded for, the total count of the codes, and a count of the number of articles they appear in can be found in Table 4.3, Table 4.4, and Table 4.5 respectively.

For the question "What developer type(s) are mentioned?" in Table 4.3 we find that all coded articles mention all developer types at least once. We find that the code `Stayer` occurs the most often when looking at count. As such, software developers staying in software development seem to get the most attention in the articles and blog posts we coded. We also find that the code `Leaver` occurs the least often when looking at count, when not including the code `N/A`. As such, software developers that leave software development seem to get the least attention in the articles and blog posts we coded.

For the question "What is the speaker's reaction to the public discourse?" in Table 4.4 we find that the code `Agree` occurs in the most articles when compared to the codes `Disagree` and `Reproduce`, although only by single extra article. When we look at the count of the codes, we find

that the code `Disagree` occurs the most often and the code `Agree` occurs the least often, although all counts are similar with `Disagree` only occurring nine more times compared to `Agree`. As such, it would seem that there is no clear overall trend in what the reaction is to the public discourse in the articles and blog posts we coded, as all codes appear a similar amount of times and in a similar amount of articles.

For the question "Is age perceived as limiting?" in Table 4.5 we find that the code `True` occurs in the most articles when compared to the codes `False` and `Unknown`, occurring in 19 out of 20 coded articles. When we look at the count of the codes over all of the articles, we again find that the code `True` occurs the most often, occurring more than twice as often as the codes `False` and `Unknown` combined. Combining the fact that the code `True` occurs the most often and in the most articles, it seems to be the case that age is overwhelmingly seen as limiting in the articles and blog posts we coded.

4.3.1 RQ1: "How is the public discourse on age in software development reproduced in popular online media?"

Final codes

We will first present the final codes we created for "How is the public discourse on age in software development reproduced in popular online media?".

Several articles contained statements such as *This rampant ageism in the technology industry causes several problems for developers both young and old* (A18) and *Silicon Valley has a reputation for favoring younger engineers, developers, and programmers* (A1) implying that there is ageism in the software industry. For statements such as these, we have created and used the code `General acknowledgement of ageism in software industry`.

Besides just a general acknowledgement of ageism, we also found that articles use statements made by authority figures, an example of which is *Facebook CEO Mark Zuckerberg famously said back in 2007 at Stanford: "Young people are just smarter."* (A1), for which we used the code `Use of authority figures`. Furthermore, results from studies are also used, as in the following paragraph, *On the heels of research that found your chances of getting hired in Silicon Valley plummet after the age of 48, a new study has found that nearly half of those already working in the industry fear getting the ax because of their advancing age* (A19). We thus also created the code `Use of related studies` for use for similar paragraphs. A commonly occurring study that is used in the coded articles and blog posts, is a demographic study, or results from a demographic study, as in the following paragraph *Six years after finishing college, 57 percent of computer science graduates are working as programmers; at 15 years the figure drops to 34 percent, and at 20 years when most are still only in their early 40s it is down to 19 percent. In contrast, the figures for civil engineering are 61 percent, 52 percent and 52 percent* (A15). As we encountered the use of demographics more often, we created the code `Demographics`.

We found paragraphs such as *Programmers need to be [constantly learning]. Skills can quickly become out-of-date or redundant. The pace in tech is relentless, with long-standing languages rendered obsolete and new languages and methodologies lionized* (A18) which refer to other articles through the use of links in the text, indicated by the square brackets in the paragraph. We also noted more explicit use of other articles such as *The TechCrunch post states, "The young understand new technologies better than the old do, and are like a clean slate." Dave Winer feels that we make the same mistakes over and over again because we are throwing away or ignoring experience by only hiring younger developers* (A16). This led us to the creation and use of the code `Use of related articles` for these and similar paragraphs. We also found that the use of forum posts occurred more often, such as in the paragraphs *Lots of links to Quora questions along the lines of Is _ too old to start as a developer* (A8) and *If all you do is 'write code' then you have to be prepared to 'write the same code' in a new paradigm several times, one commenter, ChuckMcM, wrote* (A3). We thus also added and used the code `Use of related forum posts`.

We finally created the code `Reproduction of discourse - Other` for use when none of the above codes fit the paragraph.

Counts of the codes

In Figure 4.1 we show how often the codes for "How is the public discourse on age in software development reproduced in popular online media?" have been assigned to a paragraph in the final coding of the articles and blog posts. In total 108 paragraphs have been coded using the above codes, note that this does not necessarily match the sum of the code counts as multiple codes can be assigned to a single paragraph. The code that has been assigned the most often, is `General acknowledgement of ageism in software industry`. The code `Reproduction of discourse - Other` has been assigned the least often. When we look at the paragraphs that were coded with the code `Use of authority figures`, we find that there are two authority figures that mentioned the most often, namely Norman Matloff, a professor of computer science at the University of California, and Mark Zuckerberg, CEO of Facebook. The statements of both Norman Matloff and Mark Zuckerberg used in the articles and blog posts clearly state that younger developers are preferred, as shown in *"Many programmers find that their employability starts to decline at about age 35"* (A4) and *"Young people are just smarter"* (A1) for Norman Matloff and Mark Zuckerberg respectively.

In Figure 4.2 we show in how many articles and blog posts the codes for "Which age-related aspects are reported in popular online media?" appear. The code `Use of related articles` occurs in the most articles and blog posts rather than `General acknowledgement of ageism in software industry`. The code `Use of related forum posts` occurs in the least amount of articles and blog posts rather than `Reproduction of discourse - Other`.

Combining the count of the codes over all articles and the count of the amount of articles the codes appear in, we find that the code `Use of related articles` occurs often and occurs in over half of the coded articles and blog posts. This could suggest that the articles and blog posts we coded often draw inspiration from or react to other related articles. We also find that the code `General acknowledgement of ageism in software industry` occurs often and occurs in half of the coded articles and blog posts. This again could suggest that this is something that the authors of the articles and blog posts we coded often react to in some way. Although `Use of related forum posts` occurs in only three articles, it has a total count of 17. This suggests that although forum posts are not often used in the articles and blog posts we coded, when they are present, they are used multiple times, suggesting that there is niche in which forum posts are used in articles and blog posts.

Combination with other codes

We will next present the results from coding for "How is the public discourse on age in software development reproduced in popular online media?" against "What developer type(s) are mentioned?", "What is the speaker's reaction to the public discourse?", and "Is age perceived as limiting?"

In Table 4.6 we present how often the codes for "How is the public discourse on age in software development reproduced in popular online media?" have been assigned to a paragraph combined with "What developer type(s) are mentioned?". Note that multiple codes from "What developer type(s) are mentioned?" could be assigned to paragraph. Overall, we find that the combination with the code `Stayer` occurs the most often, with this only not being the case for `Demographics`, `Use of related forum posts` and `Reproduction of discourse - Other`. The combination with the code `N/A` occurs the least often, except for with the code `Reproduction of discourse - Other`. Out of the three developer type codes, `Leavers` occurs the least often.

Overall, stayers thus seem to the be the main focus of the public discourse used in the coded articles and blog posts whereas leavers receive the least attention in the public discourse in the coded articles and blog posts. When we look at the distribution of developer types found for RQ1, against the distribution of developer types in the overall data, see Table 4.9a, we note that they appear to have a similar distribution. It could thus be the case that stayers occur more often due to the fact that they are mentioned more often in the data overall.

It is interesting to note that the results for `Demographics` do not follow the same trend as the

Code	Short description	Count	Article count
Joiner	The paragraph talks about people joining software development	107	20
Stayer	The paragraph talks about people staying in software development	152	20
Leaver	The paragraph talks about people that leave software development	73	20
N/A	The paragraph does not mention or imply any specific developer type(s)	24	20

Table 4.3: Codes for "What developer type(s) are mentioned?"

Code	Short description	Count	Article count
Agree	The paragraph agrees with the public discourse	58	16
Disagree	The paragraph disagrees with the public discourse	67	15
Reproduce	The paragraph only reproduces public discourse	63	15
N/A	The paragraph does not react to or reproduce the public discourse	58	20

Table 4.4: Codes for "What is the speaker's reaction to the public discourse?"

Code	Short description	Count	Article count
True	Age is limiting in software development in the paragraph	129	19
False	Age is not seen as limiting in software development in the paragraph	34	13
Unknown	It is not clear whether age is seen as limiting from the paragraph	17	6
N/A	The paragraph is not applicable for "Is age perceived as limiting?"	66	20

Table 4.5: Codes for "Is age perceived as limiting?"

Figure 4.1: Count for RQ1 codes

rest of the data, which is mainly used to discussed leavers. The demographics used are mostly from Stack Overflow [1] and may not be representative of the actual developer population. The results for `Use of related forum posts` also do not follow the same trend, as joiners are the main focus.

In Table 4.7 we present how often the codes for "How is the public discourse on age in software development reproduced in popular online media?" have been assigned to a paragraph combined with "What is the speaker's reaction to the public discourse?". Overall, we find that the combination with the code `Reproduce` occurs the most often, with this not being the case for `General acknowledgement of ageism in software industry` as well as `Reproduction of discourse - Other`. The combination with the code `N/A` occurs the least often, except for the combination with the codes `Use of related articles`, `Demographics`, `Use of related forum posts`, and `Reproduction of discourse - Other`. Out of the three codes besides N/A, `Leavers` occurs the least often.

When using public discourse on age(ing) in software development, the overall use seems to be neutral, i.e. reproducing the discourse, in the 20 code articles and blog posts. We again compare the distribution of the speaker's reaction for RQ1 against the overall distribution of reactions in the data, see Table 4.9b. We find that the distribution of the reactions in the overall data is far more mixed, being split between agreeing, disagreeing, and reproducing. As such, it would seem that the reactions to the public discourse for the RQ1 codes are more focused on reproducing the discourse.

For the case `General acknowledgement of ageism in software industry` it is not the case that it is mostly used for reproduction, as the speaker's agree more than any other option.

In Table 4.8 we present how often the codes for "How is the public discourse on age in software development reproduced in popular online media?" have been assigned to a paragraph combined with "Is age perceived as limiting?". We find that the combination with the code `True` occurs the most often, with this not being the case for `Demographics` and `Reproduction of discourse - Other`. The combination with the code `False` occurs the least often, except for `Use of authority figures`.

The public discourse on age(ing) in software development used in the coded articles and blog posts overwhelmingly agrees that age is as limiting. As with the developer types and the reactions to the discourse, we check to see whether the fact that the public discourse in the coded articles and blog posts overwhelmingly agrees that age is as limiting could be caused by the overall data finding age to be limiting, see Table 4.9c. We find that, although the overall data does find age to be limiting in the majority of the cases, the degree with which the public discourse finds age to be limiting, seems to be higher. As such, age seems to be overwhelmingly presented as limiting in the public discourse, more so than in the overall data.

We also find that for demographics, the reaction is split, with it being mainly unknown whether age is limiting. Furthermore, there are very few occurrences of age not being limiting, only five, in the public discourse being reproduced.

4.3.2 RQ2: "Which age-related aspects are reported in popular online media?"

Final codes

We will first present the final codes we created for "Which age-related aspects are reported in popular online media?".

We found references to not fitting in to aspects of software development due to age. In particular, social aspects were pointed out in paragraphs such as *I would walk into some interviews and I would get the impression of, Oh, whats my mom doing here? says Hill, who worked as a software developer for 10 years before taking a career break.* (A20) and *Although ageism is rare, you may feel out of place at times since most of your coworkers will be much younger than you*

[1] `https://stackoverflow.com/`

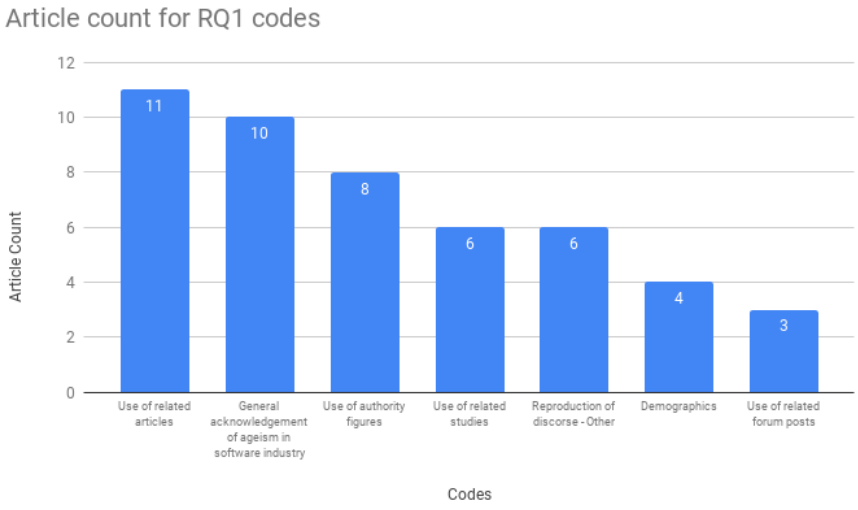


Figure 4.2: Article count for RQ1 codes

	Joiner	Stayer	Leaver	N/A
General acknowledgement of ageism in software industry	12	22	11	2
Use of authority figures	7	9	7	2
Use of related studies	7	14	5	0
Use of related articles	8	14	10	0
Demographics	6	8	9	0
Use of related forum posts	10	7	5	2
Reproduction of discourse - Other	4	3	1	4
Total	54	77	48	10

Table 4.6: Count for RQ1 codes against "What developer type(s) are mentioned?"

	Agree	Disagree	Reproduce	N/A
General acknowledgement of ageism in software industry	14	4	9	2
Use of authority figures	3	5	8	1
Use of related studies	0	1	14	0
Use of related articles	5	0	11	1
Demographics	0	3	5	3
Use of related forum posts	4	3	6	4
Reproduction of discourse - Other	0	4	3	3
Total	26	20	56	14

Table 4.7: Count for RQ1 codes against "What is the speaker's reaction to the public discourse?"

	True	False	Unknown	N/A
General acknowledgement of ageism in software industry	27	0	2	0
Use of authority figures	13	3	0	1
Use of related studies	9	0	3	3
Use of related articles	13	1	1	2
Demographics	5	0	6	0
Use of related forum posts	11	1	1	4
Reproduction of discourse - Other	3	0	0	7
Total	81	5	13	17

Table 4.8: Count for RQ1 codes against "Is age perceived as limiting?"

	Total count	RQ1 count
Joiner	107	54
Stayer	152	77
Leaver	73	48

(a) Table for total count developer type against RQ1 count

	Total count	RQ1 count
Agree	58	26
Disagree	67	20
Reproduce	63	56

(b) Table for total count reaction against RQ1 count

	Total count	RQ1 count
True	129	81
False	34	5
Unknown	17	13

(c) Table for total count is age limiting against RQ1 count

Table 4.9: Tables for total counts against RQ1 counts

are, he wrote. This comes up more often in social situations than in technical ones. (A6). As such, we created the code `Social challenges` for paragraphs such as these.

In addition to social challenges being described, we find paragraphs including *The harsh reality is that in the tech world, companies prefer to hire young, inexperienced, engineers.* (A17) and *And sure enough, I found people of all ages who are worried that theyre too old to learn to code and get hired as a developer* (A8) that point to professional challenges due to age in software development. Thus, we created the code `Professional challenges`.

We also found references to what can be considered the career path of a software developer. For example, the paragraphs *One ancient ie 35 year old developer advised that the best thing to do is plan to retire by the time you're 40* (A3) as well as *Other programmers are inevitably promoted to management. I know, you're rolling your eyes. "A career in software development doesn't necessarily prepare you to be a great manager," you're saying. Guess what? Neither do management training courses.* (A4) point to two possible paths in a software developers career. Accordingly, we created the code `Career path` to be used for paragraphs such as these.

Lastly, we created the code `Reported aspects - Other` for use when none of the above codes fit the paragraph.

Count of the codes

In Figure 4.3 we show how often the codes for "Which age-related aspects are reported in popular online media?" have been assigned to a paragraph in the final coding of the articles and blog posts. In total 142 paragraphs have been coded using the above codes, note that this does not necessarily match the sum of the code counts as multiple codes to be assigned to a single paragraph. The code that has been assigned the most often, is `Professional challenges`. The code `Reported aspects - Other` has been assigned the least often.

In Figure 4.4 we show in how many articles and blog posts the codes for "Which age-related aspects are reported in popular online media?" appear. Similar to the count of the occurrences of the codes, the code that appears in the most amount of articles and blog posts, is `Professional challenges`, however, the code that appears in the least amount of articles and blog posts is `Social challenges` rather than `Reaction - Other`.

Combining the count of the codes over all articles and the count of the amount of articles the codes appear in, the code `Professional challenges` occurs both the most often and in the most articles and blog posts, with it only not being mentioned in a single coded article or blog post. When we look at the code `Career path`, we also note that is occurs in 14 coded articles or blog posts, occurring 54 times in total, thus being present on average almost four times in each article mentioning the career path of software developers. Professional challenges and the career path of software developers are mentioned so often, it suggests that they are numerous or seen as important issues for the coded articles and blog posts.

When we look at the codes `Social challenges` and `Reported aspects - Other`, we find that they both do not occur often when compared to the codes `Professional challenges` and `Career path`, and occur in fewer coded articles and blog posts, but still in almost half or more than half of the coded articles and blog posts. This could suggest that although social challenges and other reported aspects are well known, being present in around half of the articles, they are seen as less important or are less numerous than professional challenges or aspects of the career path of software developers, thus having a lower total count.

Combination with other codes

We will next present the results from coding for "Which age-related aspects are reported in popular online media?" against "What developer type(s) are mentioned?", "What is the speaker's reaction to the public discourse?", and "Is age perceived as limiting?"

In Table 4.10 we present how often the codes for "Which age-related aspects are reported in popular online media?" have been assigned to a paragraph combined with "What developer type(s) are mentioned?". Note that multiple codes from "What developer type(s) are mentioned?" could

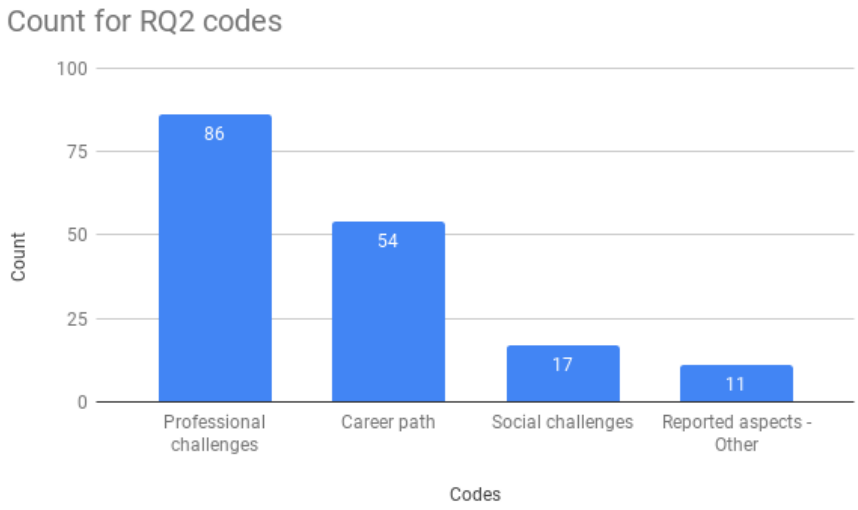


Figure 4.3: Count for RQ2 codes

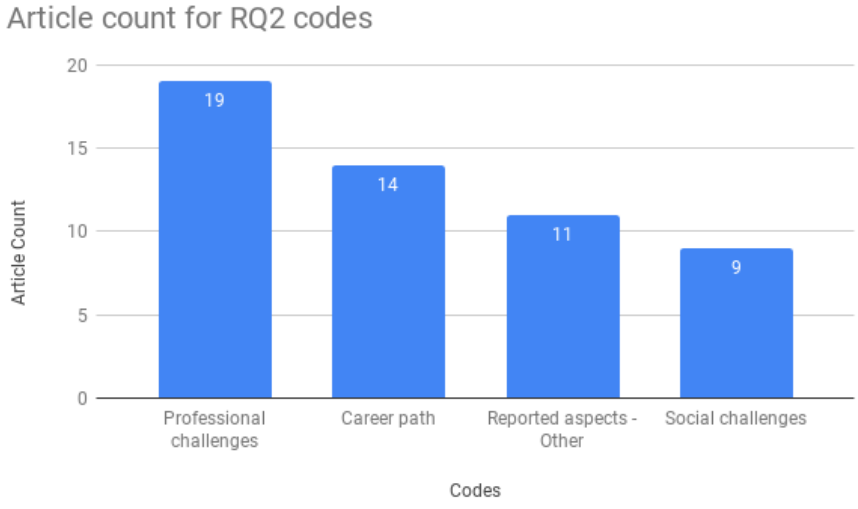


Figure 4.4: Article count for RQ2 codes

be assigned to paragraph. Overall, we find that the combination with the code `Stayer` occurs the most often, with almost double the amount of occurrences than the next highest, and the combination with the code `N/A` occurring the least often. Out of the three developer type codes, `Leavers`, as in the combination with "How is the public discourse on age in software development reproduced in popular online media?", occurs the least often.

Overall, stayers receive the most attention when looking at age-related issues in the coded articles and blog posts. We compare this with the overall distribution of the developer types in Table 4.13a. We find that the ordering of the developer types is the same, but there seems to be a slightly larger focus on stayers than in the distribution of the complete data.

Looking at `Social challenges`, we find that they seem to be aimed more at joiners, suggesting they are more of an issue when joining software development later in life. Joiners are also mentioned less often with the code `Career path`, with leavers being mentioned more often. This suggests more career path related issues occur further in ones career in software development.

In Table 4.11 we present how often the codes for "Which age-related aspects are reported in popular online media?" have been assigned to a paragraph combined with "What is the speaker's reaction to the public discourse?". Overall, we find that the combination with the code `Agree` occurs the most often, except for the code `Reported aspects - Other`. The combination with the code `Disagree` occurs the least often, besides for the combination with the code `Career path`, and the code `Reported aspects - Other` for which the combination is the most common.

The age-related aspects mentioned in the coded articles and blog posts are thus agreed with. We compare this with the overall distribution of the reaction types in Table 4.13b. We find that the ordering of the reaction types differs from the ordering in the complete data, with agreeing being more common when presenting age-related aspects. This suggests that the speakers agree that the age-related aspects in the public discourse are indeed present for software developers.

In Table 4.12 we present how often the codes for "Which age-related aspects are reported in popular online media?" have been assigned to a paragraph combined with "Is age perceived as limiting?". Overall, we find that the combination with the code `True` occurs the most often, except for the code `Reported aspects - Other`. The combination with the code `Unknown` occurs the least often, except for with `Reported aspects - Other`.

As the combination with the code `True` occurs the most often, it would seem that the majority of the age-related aspects mentioned perceive age as limiting for all codes besides `Reported aspects - Other`. When we compare this against the overall distribution of whether age is perceived as limiting, see Table 4.13c, we find that they follow a similar distribution. As such, it could be that the majority of age-related aspects are seen as limiting due there being more paragraphs that perceive age as limiting overall.

4.3.3 RQ3: "What is the reaction to the public discourse in the popular online media?"

Final codes

We will first present the final codes we created for "What is the reaction to the public discourse in the popular online media?".

When looking at the reaction to the public discourse, we found examples of personal experiences being mentioned such as *During my tech days, I hired several programmers who were over 50. They were the steadiest performers and stayed with me through the most difficult times* (A12) and *When I took my first job as a developer, I was the oldest person in the company and the most junior. This is a very odd experience, especially since before I switched jobs, I had held managing positions and had previously been in charge of over 60 people* (A13). This led to the creation of the code `Personal experiences`.

Besides reactions based on personal experiences, we also found that speculation is used to react to certain thoughts in the public discourse, for example in *Of course, the data isn't an absolute indication of the tech workforce in Silicon Valley or any other area of the world it just provides a bit more evidence to the argument that younger developers are in higher demand* (A1) and

	Joiner	Stayer	Leaver	N/A
Social challenges	13	11	3	0
Professional challenges	33	68	21	3
Career path	12	31	26	2
Reported aspects - Other	3	10	1	1
Total	61	120	51	6

Table 4.10: Count for RQ2 codes against "What developer type(s) are mentioned?"

	Agree	Disagree	Reproduce	N/A
Social challenges	10	0	3	3
Professional challenges	28	11	25	22
Career path	15	15	10	14
Reported aspects - Other	0	8	1	2
Total	53	34	39	41

Table 4.11: Count for RQ2 codes against "What is the speaker's reaction to the public discourse?"

	True	False	Unknown	N/A
Social challenges	13	1	0	2
Professional challenges	68	5	0	13
Career path	33	10	4	7
Reported aspects - Other	0	8	1	2
Total	114	24	5	24

Table 4.12: Count for RQ2 codes against "Is age perceived as limiting?"

	Total count	RQ2 count
Joiner	107	61
Stayer	152	120
Leaver	73	51

(a) Table for total count developer type against RQ2 count

	Total count	RQ2 count
Agree	58	53
Disagree	67	34
Reproduce	63	39

(b) Table for total count reaction against RQ2 count

	Total count	RQ2 count
True	129	114
False	34	24
Unknown	17	5

(c) Table for total count is age limiting against RQ2 count

Table 4.13: Tables for total counts against RQ2 counts

Developers often skew young because older staff can grow tired of relearning their skills each time a new platform comes out. Second, and more importantly, companies frequently hire younger, inexperienced programmers to perform the same work for a cheaper salary (A3). As such, we created the code `Speculation`.

We finally created the code `Reaction - Other` for use when none of the above codes fit the paragraph.

Count of the codes

In Figure 4.5 we show how often the codes for "What is the reaction to the public discourse in the popular online media?" have been assigned to a paragraph in the final coding of the articles and blog posts. In total 95 paragraphs have been coded using the above codes, note that this does not necessarily match the sum of the code counts as multiple codes to be assigned to a single paragraph. The code that has been assigned the most often, is `Speculation` with the code `Reaction - Other` having been assigned the least often.

In Figure 4.6 we show in how many articles and blog posts the codes for "What is the reaction to the public discourse in the popular online media?" appear. Similar to the count of the occurrences of the codes, the code that appears in the most amount of articles and blog posts, is `Speculation` with the code `Reaction - Other` appearing in the least amount of articles and blog posts.

Combining the count of the codes over all articles and the count of the amount of articles the codes appear in, the code `Speculation` appears both the most often, almost twice as often as `Personal experiences`, and in the most articles. The average amount of paragraphs containing speculation being almost five per article or blog post containing speculation. Although the code `Personal experiences` occurs less often than `Speculation`, it still occurs 32 times, and is present in half of the coded articles and blog posts. The average amount of paragraphs containing personal experiences to react to the public discourse being three per article or blog post containing personal experiences. Overall, the use of speculation and personal experiences both seem to be a common way to respond to the public discourse in the articles and blog posts that we coded, with the use of speculation being more common.

Combination with other codes

We will next present the results from coding for "What is the reaction to the public discourse in the popular online media?" against "What developer type(s) are mentioned?", "What is the speaker's reaction to the public discourse?", and "Is age perceived as limiting?"

In Table 4.14 we present how often the codes for "What is the reaction to the public discourse in the popular online media?" have been assigned to a paragraph combined with "What developer type(s) are mentioned?". Note that multiple codes from "What developer type(s) are mentioned?" could be assigned to paragraph. Overall, we find that the combination with the code `Stayer` occurs the most often, although only by a single result as well as only occurring the most often for the code `Speculation`, and the combination with the code `N/A` occurring the least often. Out of the three developer type codes, `Leavers`, as in the combination with "How is the public discourse on age in software development reproduced in popular online media?" and "Which age-related aspects are reported in popular online media?", occurs the least often.

When reacting to the public discourse, stayers are mentioned the most often, followed closely by joiners. In Table 4.17a, we compare the distribution of the developer types found for RQ3 and distribution of the whole data. We find that while the ordering is the same, joiners seem to be mentioned at a higher ratio when compared to the distribution of the whole data. As such, joiners seem to receive more attention than could be expected from the trend of the data as a whole, when looking at reactions to the public discourse.

Personal experiences seem to mostly be used to talk about joiners, with only few personal experiences mentioning leavers. There is almost the same amount of speculation about joiners, however the use of speculation seems to be spread more evenly across both joiners and leavers, with most speculation focusing on stayers.

Figure 4.5: Count for RQ3 codes

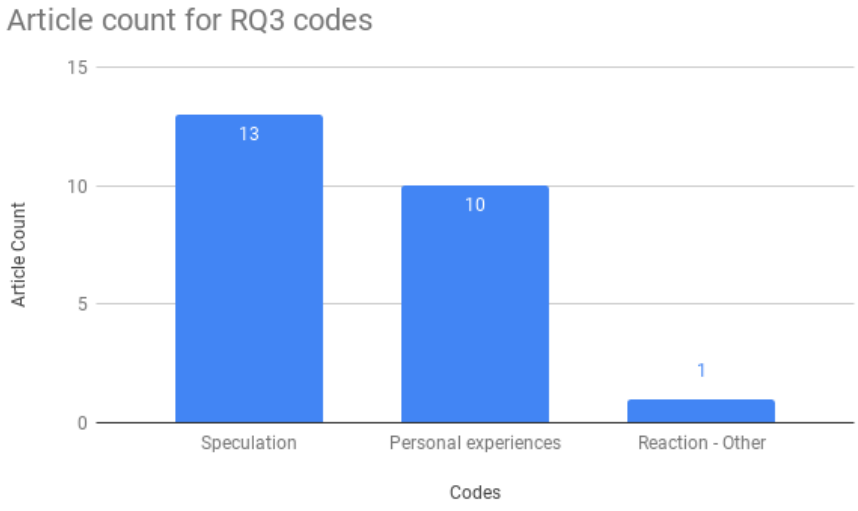


Figure 4.6: Article count for RQ3 codes

In Table 4.15 we present how often the codes for "What is the reaction to the public discourse in the popular online media?" have been assigned to a paragraph combined with "What is the speaker's reaction to the public discourse?". Overall, we find that the combination with the code `Disagree` occurs the most often, although only by two results, and with it not occurring the most often for the code `Speculation`. The combination with the code `N/A` occurs the least often.

Thus, we find that the reaction to the public discourse is mixed, reactions disagreeing to the public discourse being slightly more common, suggesting that there is no common thought on the public discourse. We compare this to the overall distribution of reactions in the data, see Table 4.17b. We find that the reactions are more mixed when comparing agree and disagree, with the number of occurrences being closer for RQ3. However, the biggest difference is that there are much fewer occurrences of reproduction. As such, the focus for RQ3 is much more on reacting to the public discourse.

In Table 4.16 we present how often the codes for "What is the reaction to the public discourse in the popular online media?" have been assigned to a paragraph combined with "Is age perceived as limiting?". We find that the combination with the code `True` occurs the most often, and the combination with the code `Unknown` occurring the least often. In Table 4.17c we compare the distribution of whether age is seen as limiting for RQ3, and in data as a whole. We find that the distribution for RQ3 seems to follow a similar distribution as the data as a whole. It could thus be the case that age is seen as limiting for RQ3 due to the fact that the majority of time in the data, age is seen as limiting, rather than statements that perceive age as limiting generating a greater response.

	Joiner	Stayer	Leaver	N/A
Personal experiences	25	8	4	1
Speculation	24	42	22	3
Reaction - Other	0	0	0	2
Total	49	50	26	6

Table 4.14: Count for RQ3 codes against "What developer type(s) are mentioned?"

	Agree	Disagree	Reproduce	N/A
Personal experiences	13	14	3	2
Speculation	28	27	7	0
Reaction - Other	0	2	0	0
Total	41	43	10	2

Table 4.15: Count for RQ3 codes against "What is the speaker's reaction to the public discourse?"

	True	False	Unknown	N/A
Personal experiences	18	6	2	6
Speculation	35	13	4	10
Reaction - Other	1	0	0	1
Total	54	19	6	17

Table 4.16: Count for RQ3 codes against "Is age perceived as limiting?"

	Total count	RQ3 count
Joiner	107	49
Stayer	152	50
Leaver	73	26

(a) Total count developer type against RQ3 count

	Total count	RQ3 count
Agree	58	41
Disagree	67	43
Reproduce	63	10

(b) Table for total count reaction against RQ3 count

	Total count	RQ3 count
True	129	54
False	34	19
Unknown	17	6

(c) Table for total count is age limiting against RQ3 count

Table 4.17: Tables for total counts against RQ3 counts

4.4 Further analysis of coding results

In this section, we will present the findings from the analysis of the articles from the perspective of employability strategies and the subcoding performed on the results for RQ2: "Which age-related aspects are reported in popular online media?".

4.4.1 Employability strategies

We will first present the results from the card sorting performed by two researchers for employability strategies. The resulting mind map from the card sorting can be found in Figure 4.7. The mind map has two main hierarchies relating to employability strategies for older developers. The first is based on individuals and strategies they can adopt, whereas the second is based on companies and aspects that effect older developers in a company.

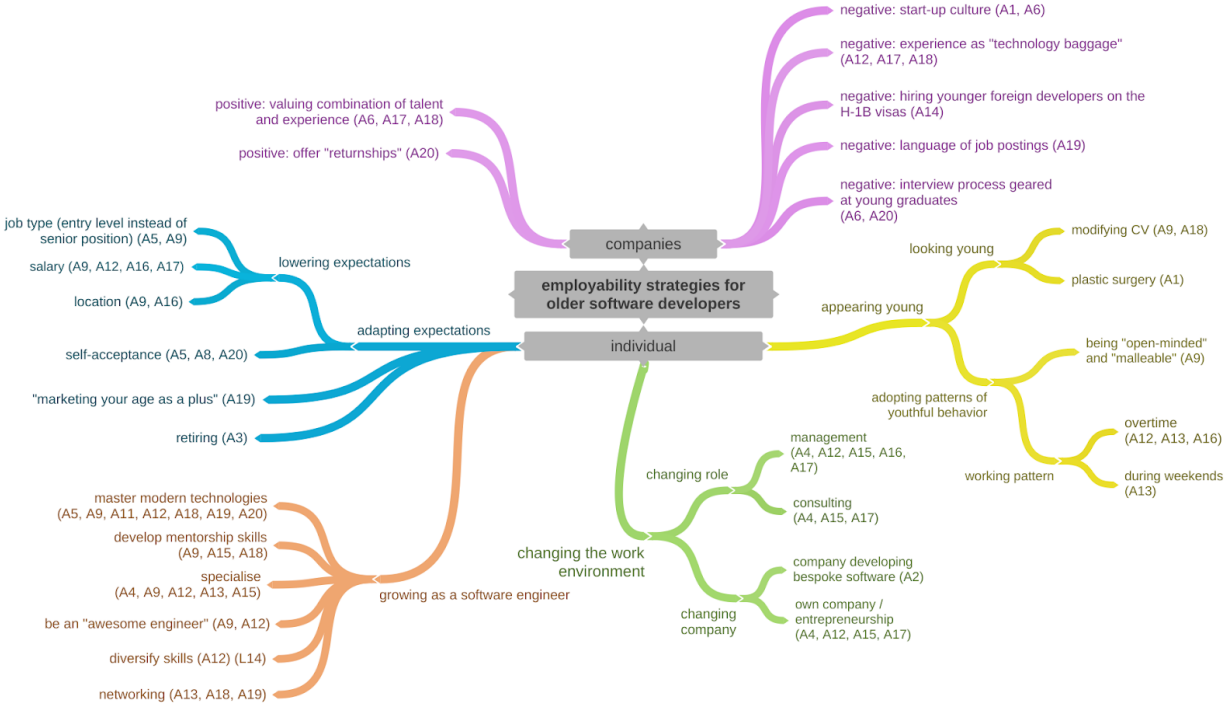


Figure 4.7: Employability strategies mind map

We will first discuss the hierarchy for individuals. In the 20 articles and blog posts we found statements such as *you're old, get over it* (A5) and *One even suggested that the solution to being "too old" at 30 was simple: developers should retire at 40* (A3). These strategies all point towards changing what to expect as an older software developer. As such, we created a group of strategies called `Adapting expectations`.

We also created a group of strategies called `Growing as a software engineer`. With this strategy, we include strategies mentioned in the articles such as `Mastering modern technologies`, an example of which is *Keep your skills current. This means keeping up to date with the latest trends in computing, programming techniques, and languages, and adapting to change* (A12), and `Develop mentorship skills`, such as mentioned by *Leverage your experience. Instead of being "the older guy who needs a higher salary," be the "guy with a lot of experience and great mentor potential"* (A9).

Besides growing as a software engineer and adapting expectations, we also found that articles mentioned strategies such as moving to management or consultancy, *Move up the ladder into management, architecture, or design; switch to sales or product management; or jump ship and*

become an entrepreneur (A17), as well as becoming an entrepreneur, *Become an entrepreneur. Despite what some investors say, older age is an advantage in the startup world* (A12). As such, we created a group of strategies called `Changing the work environment`.

Lastly, we found statements in the articles such as *Remove graduation dates and older positions from your resume. You might be 45, but if you look like you're in 30s, you can use that to your advantage* (A9), which point to being, or seeming to be, youthful as an employment strategy. Furthermore, adopting younger persons work patterns such as overtime was also mentioned, *For all of my other jobs, when my day was over, I left and went home to try and not think about work. Now in tech, I often dont go home but go to meetups straight after work a couple of times a week* (A13). This led us to create a group of strategies called `Appearing young`.

Next, we will discuss the findings in the company hierarchy, which we have split between positive and negative aspects. On the positive side of things we found by companies, we find the aspects `Offer returnships`, as described in *Some companies are beginning to offer returnships internship programs to attract talented job seekers who have taken career breaks and need to revamp their skills* (A20), and `Valuing combination of talent and experience`, as we find here *But there are many companies of all size that recognize how potent the combination of talent and experience can be* (A6).

On the negative side of aspects found in companies, we find that experience is not always seen as postive, as with the code `Experience as ‘‘technology baggage"` from *The young understand new technologies better than the old do, and are like a clean slate: They will rapidly learn the latest coding methods and techniques, and they dont carry any technology baggage* (A12). We also find that the hiring process can be negative for older developers, as with the `Language of job posts` as well as `Interview process geared at young graduates` such as in and *Removing terms like recent graduate and digital native could help encourage older professionals to apply for these positions* (A19) and *The main reasons Google engineers skew slightly younger, on average, are that Google was much smaller 10 years ago; many of its hires are made at the entry level; and for more senior positions, many older engineers are too well-established in their companies to transfer* (A6) respectively.

Lastly, we also encounter statements such as `Some may feel "older" employees won’t fit in with the startup-like culture at some tech companies  older employees may have children they’d like to spend more time with, for instance` (A1) and *One major benefit of hiring an H-1B is that she is a de facto indentured servant, trapped with her employer, who then need not worry that she might jump ship for another firm, a huge advantage* (A14). These point at the aspects `Startup-culture` and `Hiring younger foreign developers on the H1B visa` respectively that negatively affect the employability of older developers.

4.4.2 Subcodes

Next, we present the subcodes we created for RQ2: “Which age-related aspects are reported in popular online media?”.

In Table 4.18 we present the subcodes, and example of the subcode from the articles and blog posts, and the count of the subcode, for `Social challenges`. In total, we created four subcodes that relate to social challenges that older developers face, with the most occurring subcode being `Fitting in with young people`.

In Table 4.19 we present the subcodes, and example of the subcode from the articles and blog posts, and the count of the subcode, for `Professional challenges`. We created seven subcodes for professional challenges that developers face, with the subcode `Younger developers being cheaper/preferred` being mentioned the most often. It is interesting to note that all but one of the subcodes somehow suggest that age(ing) is negative for software developers, with only `Challenges for younger developers` presenting a professional challenge that is not related to older developers.

In Table 4.20 we present the subcodes, and example of the subcode from the articles and blog posts, and the count of the subcode, for `Career path`. We created a total of seven subcodes related to the career path of software developers. Of those seven, the most common subcode

is `Declining employability of older developers`, suggesting that when talking about career paths of developers, their declining employability when ageing is mentioned the most often. It is interesting to note that the codes `Advice on how to stay a software developer as you get older` and `Showing there are older developers` were created, suggesting that it is possible to stay a software developer as one gets older.

In Table 4.21 we present the subcodes, and example of the subcode from the articles and blog posts, and the count of the subcode, for `Reported aspects - Other`. For the code `Reported aspects - Other`, we created 4 subcodes, with the most common subcode being `More knowledgeable/Experience`. It is interesting to note that all these subcodes show age-related aspects that can be seen as positive in some way.

Code	Example	Count
Fitting in with young people	*Although ageism is rare, you may feel out of place at times since most of your coworkers will be much younger than you are, he wrote* (A6)	8
Family obligations that take up time	*This can be a big challenge for people breaking into the industry when older because some of us have families and less time to attend events* (A13)	5
Create a network of contacts	*Finally, wherever you work now, however many winters youve seen, network. Build relationships, make yourself known. Make friends and establish yourself as a valuable team member* (A18)	1
Needing to be humble	*Hill acknowledged that returning to work took "a big dose of humility"* (A20)	3

Table 4.18: Subcodes for `Social challenges`

Code	Example	Count
Fewer senior positions available	*The main reasons Google engineers skew slightly younger, on average, are that Google was much smaller 10 years ago; many of its hires are made at the entry level; and for more senior positions, many older engineers are too well-established in their companies to transfer* (A6)	1
Younger developers are cheaper/preferred	*The more likely reason, he believes, is that experienced developers cost too much money.* (A7)	29
Age being a cause for concern	*Do you worry about finding a job? Surely you don't, if you can code. It turns out that even developers have something to worry about if you're 45 or older* (A2)	11
Pressure to stay relevant and keeping up with new tech	*Make sure youre skilled in the latest technology, and are prepared for where your field could be heading* (A19)	20
Challenges for younger developers	*And it isn't any better for young developers either. The ageism in tech has also created an unhealthy attitude toward new, younger developers.* (A18)	2
Experience being disadvantageous	*When it comes to job hunting, a common concern is not having enough experience. In tech, the fear riddling job seekers and holders alike is the opposite. They worry they have too much experience* (A18)	12
Long hours in software development	*If your company requires long hours, and you do not want to work those hours, age is not the issue; the hours are the issue* (A16)	1

Table 4.19: Subcodes for `Professional challenges`

Code	Example	Count
Employers expectations from older developers	*Because employers assume you want more responsibility and/or more money than someone who is more junior, the bar for hiring you is generally much higher* (A9)	4
Changing jobs	*By 30, most of us have decided that we want to do something else: management, quantitative finance, or startup entrepreneurship, he wrote* (A3)	11
Relearning skills as one gets older	*Developers often skew young because older staff can grow tired of relearning their skills each time a new platform comes out* (A3)	4
Advice on how to stay a software developer as you get older	*Be open to more junior positions and/or flat hierarchies, especially if you're entering a new software area* (A9)	8
Declining employability of older developers	*The harsh reality is that if you are middle-aged, write computer code for a living, and earn a six-figure salary, youre headed for the unemployment lines. Your market value declines as you age and it becomes harder and harder to get a job* (A12)	14
Showing there are older developers	*For example, one of my friends was a high school French teacher in her 50s. After taking some free online university courses, she got a job as a software engineer at Apple* (A8)	4
Salary increases stop at a certain age	*After 50, the mean salary of engineers was lowerby 17% for those with bachelors degrees, and by 14% for those with masters degrees and PhDsthan the salary of those younger than 50* (A17)	2

Table 4.20: Subcodes for `Career path`

Code	Example	Count
Proficiency in older technology	*The industry turns on a dime, but is slow to retire proven technology. It is highly likely that you will still be able to earn some decent coin in the technology you know and love even after a few decades* (A15)	1
Older developers are steady performers	*During my tech days, I hired several programmers who were over 50. They were the steadiest performers and stayed with me through the most difficult times* (A12)	1
More knowledgeable/-Experience	*Workers 50+ tend to be self-starters, know how to get the job done, and don't need as much handholding as those with less experience. A great benefit to being older is that you have a good deal of knowledge and leadership ability* (A19)	8
Being able to be mentor	*They also benefit from a diverse team because they get a mentor that can help them learn and grow* (A18)	2

Table 4.21: Subcodes for `Reported aspects - Other`

4.5 Forum Posts

In this section, we will present the collection of the forum posts, the selection of similarity method to use to find similar forum posts to selected coded paragraphs, and the results from the analysis of the forum posts.

4.5.1 Similarity program settings

We will first present the results from the collection of the forum posts. After finding the ten Hacker News posts as shown in Section 4.1 that discuss an article included in our study, we used the Hacker News API [2] to retrieve all the comments. For the comments, we stored the complete JSON response from the Hacker News API, which normally contains:

- by: The author of the comment
- id: The id of the comment
- kids: The comments responding to this comment
- parent: The comment, or story, that this comment is responding to
- text: The text of the comment
- time: The date and time the comment was posted
- type: The type of post, only comment in this case

This amounted to a total of 1726 comments being retrieved and stored.

After the collection of the forum posts, we performed the normalisation of the forum posts and the coded paragraphs. Before calculating the similarity between selected coded paragraphs and forum posts, we needed to select the settings we would use to calculate similarity. In Section 3.7.1, we could not find whether to use n-gram or w-shingles and what size of n or w we should use based on literature, and thus decided on running our program with different settings to test which would be suitable for us. We ran our program with the following settings:

- n-grams of size $n \mid n \in \{3, 4, 5\}$ with cosine similarity with the BM15 weighting scheme with $k = 1.5$
- w-shingles of size $w \mid w \in \{1, 2, 3\}$ with cosine similarity with the BM15 weighting scheme with $k = 1.5$

We first tested whether to select n-grams or w-shingles. In Figure 4.8 we show the histograms of the resulting similarity scores with n-grams of the forum posts for the coded paragraph:

`What puts an age limit in the programming field is not the age itself, it is the ability adapt and stay tuned to the latest technologies. If a programmer is not willing to keep an eye on the newest programming languages and latest updates, his career will end at a certain age, maybe before he even reaches 40.`

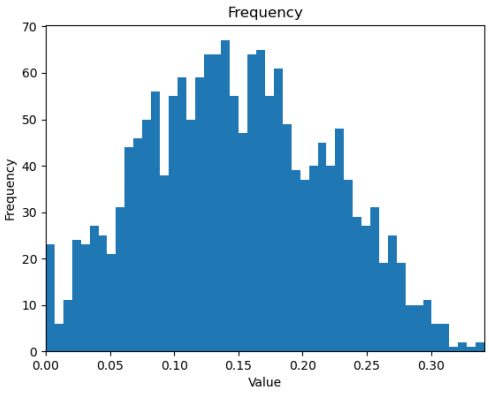
We find that the distribution of the similarity scores for $n = 3$ somewhat approach a normal distribution at first inspection, whereas the distributions for $n = 4$ and $n = 5$ do not. As such, we select n-grams with $n = 3$ as a possible candidate for finding similar forum posts.

In Figure 4.9 we show the histograms of the resulting similarity scores with w-shingles of the forum posts for the same coded paragraph. Note that there is no figure for $w = 3$, as the similarity scores all became 0 and this setting is thus discarded. We find that the distribution of the similarity scores for $w = 1$ and $w = 2$ do not approach a normal distribution. As such, we do not select any w-shingles settings as a possible candidate for finding similar forum posts.

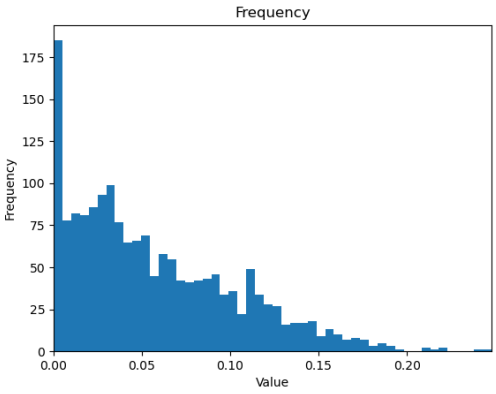
Before continuing with the single candidate setting we found, in which the distribution of the similarity scores somewhat approaches a normal distribution, we plot the distribution of the similarity scores of more coded paragraphs, to check whether the distribution also approaches a normal distribution at first inspection for those. We also plotted the distribution of of the similarity scores of the forum posts for the following paragraphs:

- `Be an expert in something. This can be a language (Clojure, Java, C, whatever), a field (Database System Design, Algorithm Design, Machine Learning, etc.), or a specific type of software (Fraud Detection Systems,`

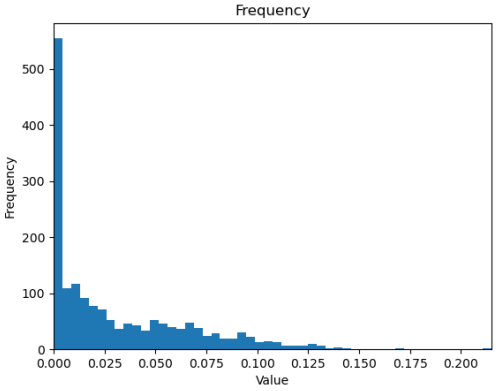
[2] https://github.com/HackerNews/API



(a) Histogram n-gram size 3



(b) Histogram n-gram size 4



(c) Histogram n-gram size 5

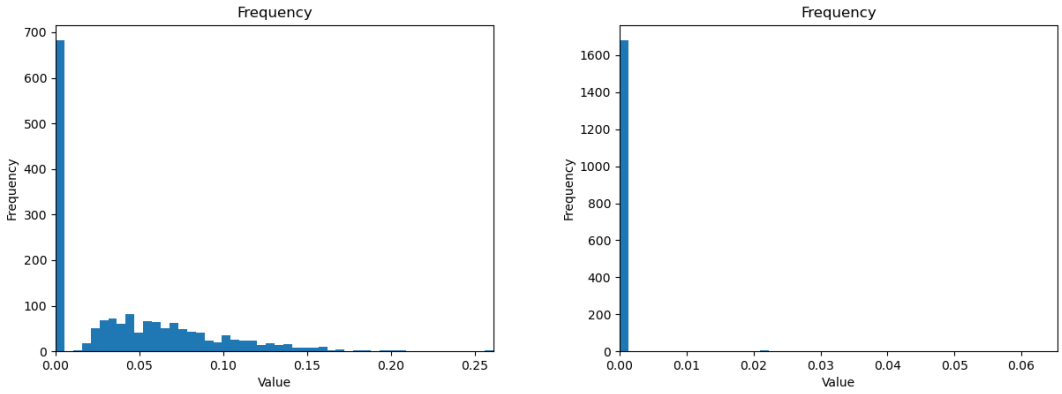
Figure 4.8: Histograms similarity scores n-gram

`Recommendation Engines, etc.). A lot of these things stay around for a decade or even longer, so if you're an expert in something, you'll be in high demand. Two caveats: 1) you should like your area of expertise, otherwise you won't be happy and 2) over time, things change, so this is a good 5-10 year plan, but not a great 30-year plan. You should pursue new areas of expertise when your current area is starting to go out of style (but don't wait until it's finished going out of style)`

- `If you decide to stay in software development, you have decided to make some maximum dollar amount in your city. If you want more money, you need to get into management`
- `Any job can be classified as a dead-end job if you dont keep on learning new skills and practice. If you want to grow and last longer in your field of interest, all you have to do is to learn how to adapt and stay abreast of the latest technologies while increasing your knowledge`

In Figure 4.10 we present the histograms of the distributions of the similarity scores with the three extra paragraphs. Again we find that the distributions somewhat approach a normal distribution. As such, we continue with the following setting:

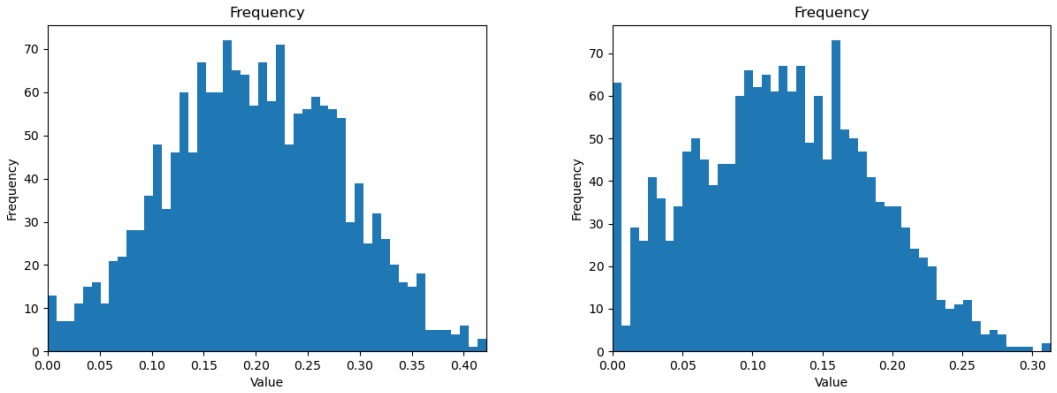
- n-grams of size $n = 3$ with cosine similarity with the BM15 weighting scheme with $k = 1.5$



(a) Histogram w-shingles size 1

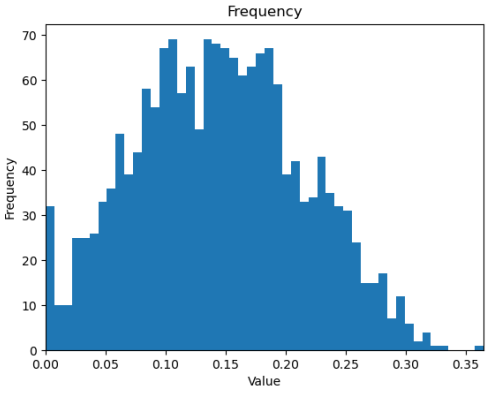
(b) Histogram w-shingles size 2

Figure 4.9: Similarity scores w-shingles



(a) Histogram for `Be an expert ...`

(b) Histogram for `If you decide ...`



(c) Histogram for `Any job can ...`

Figure 4.10: Histograms using n-grams with $n = 3$

4.5.2 Selection of most similar forum posts

After the selection of the setting we use for the calculation of the similarity score, we need to the select the highest scoring similar forum posts for certain coded paragraphs. The first step we performed, as explained in Section 3.7.2, is the checking of normality of the distribution of the similarity scores for the selection of the most similar forum posts.

To check whether the distributions of the similarity scores between forum posts and coded paragraphs follows a normal distribution, we plotted the Q-Q plots, and used the following tests:

- Shapiro-Wilk test
- Anderson-Darling test

We will first look at the the Q-Q plots of the four paragraphs used in Section 4.5.1. From the Q-Q plots, which can be found in Figure 4.11, we can see that the distributions do not seem to follow a normal distribution as the tails differ from a normal distribution with more results than expected. However, we proceed with the normality test to be certain.

In Table 4.22 we present the test scores and the p-value for the normality tests run on the distribution of the similarity scores. For all of the tests run on the data, and for all similarity score distributions, we reject the null hypothesis, that the data is from a normal distribution. As such, we will proceed by selecting the top five similar comments as discussed in Section 3.7.2.

4.5.3 Analysis of forum posts

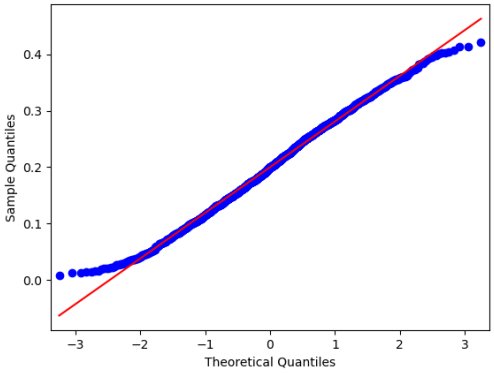
We next present and shortly discuss the results from coding forum posts for selected paragraphs for employability strategies.

For the most commonly occurring three categories of the employability strategies, namely `Master modern technologies`, `Move to management`, and `Specialise`, we selected a total of 17 coded paragraphs, eight for master modern technologies, five for move to management, and four for specialise, for which we find similar forum posts. For these 17 coded paragraphs, we found the five most similar forum posts as described in Section 3.7.2, resulting in 85 forum posts being assigned to the most similar coded paragraph.

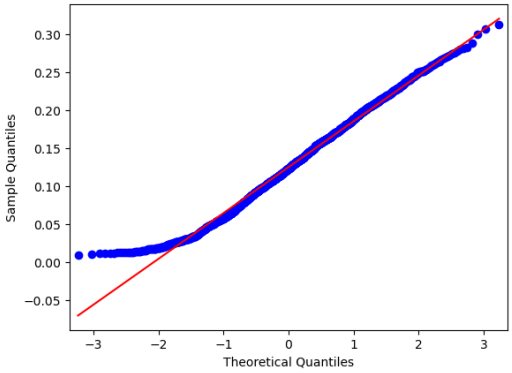
In Table 4.23 we present the results from coding the 85 forum posts. We find that a total of 55 out of the 85 forum posts were marked as relevant to the coded paragraph. Out of the 55 forum posts marked relevant, 33 were coded as agreeing with the coded paragraph, and 22 as disagreeing.

For `Master modern technologies` the forum posts disagree more often than agree with the paragraphs, although the reactions are split more evenly than for the other two categories. This could indicate that mastering modern technologies is a controversial category. When we look at the coded forum posts, we find that the disagreeing comments bring up aspects such there being a "cultural mismatch" that can not be overcome or "language agnostic" skills being more important. The comments that agree with the strategy of mastering modern technologies stating that technology will be replaced and you thus need to keep up, or that learning these technologies help to build a core understanding of software development.

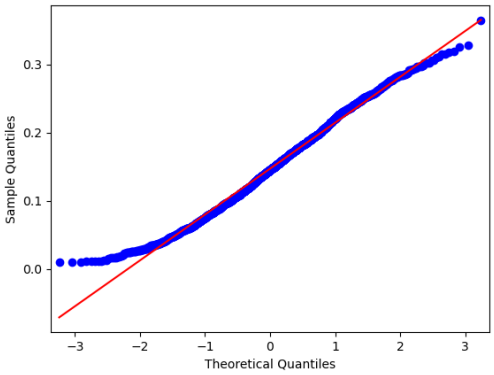
For `Move to management` and `Specialise` the reactions in the forum posts agree more often than disagree. Moving to management as well as specialising in certain technologies were seen as "safe", areas where you are less likely to lose your job, as well as ways to proceed in ones career, although software developers may not be prepared for a move to management. It was also recognised that a move to management could help break the salary ceiling that some felt was present when sticking to a programming job.



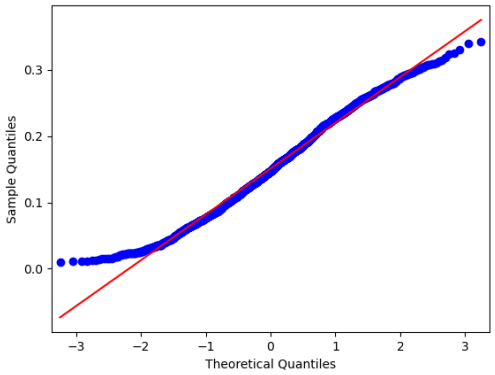
(a) Plot for `Be an expert ...`



(b) Plot for `If you decide ...`



(c) Plot for `Any job can ...`



(d) Plot for `What puts an ...`

Figure 4.11: Q-Q Plots

	Shapiro-Wilk		Anderson-Darling	
`Be an expert ...`	0.988	$p < 0.05$	1.522	$p < 0.05$
`If you decide ...`	0.995	$p < 0.05$	3.163	$p < 0.05$
`Any job can ...`	0.989	$p < 0.05$	3.587	$p < 0.05$
`What puts an ...`	0.988	$p < 0.05$	4.129	$p < 0.05$

Table 4.22: Normality test scores and p-value

	Relevant	Not Relevant	Agree	Disagree
Master modern technologies	27	13	11	16
Move to management	16	9	14	2
Specialise	12	8	8	4

Table 4.23: Coding results for selected employability strategies

Chapter 5

Discussion

In this chapter we discuss the results found in the previous literature review, summarised in Chapter 2, and the results from coding the 20 articles and blog posts and the analysis of forum posts in Chapter 4 to form answers for our research questions. Furthermore, we combine the answers of our research questions to create an overview of the public discourse on age(ing) in software development in popular online media. Lastly, we reiterate valid suggestions made in previous research on how to counter issues on age(ing) in software development found in the public discourse.

5.1 Use of Public Discourse

We first discuss our findings on the use of public discourse to answer the research question RQ1: "How is the public discourse on age in software development reproduced in popular online media?", and what implications our findings could have.

In Section 4.3.1 we found that the most common type of public discourse that is being reproduced in the online articles and blogs posts are general acknowledgements of ageism in software engineering and related articles. We also found that when the articles and blog posts make use of authority figures, Norman Matloff and Mark Zuckerberg appear the most often. Both of these made statements suggesting that younger developers are preferred. When we look at the uses of general acknowledgements of ageism in software engineering, related articles, authority figures in the articles and blog posts, they all see age as limiting in most of their uses. Indeed, we found that for the majority of statements, age is found to be limiting for the majority of their uses.

This leads on to our findings on whether age is perceived as limiting in the public discourse. We found that this differs for the public discourse compared to the trend in the complete data, with there being an even larger emphasis on age being limiting and fewer results that age is not limiting. Demographics was the exception for this, as the majority of uses do not specify whether age is limiting or not, suggesting a more neutral use than the other codes. The greater emphasis on age being limiting in public discourse that is reproduced in popular online media could imply that there is simply more public discourse available that states that age is limiting in software development. Another reason why such discourse is used more often, could just be that negative news attracts more attention. Research by Haskins et al. [20] and Zillmann et al. [50] found that the majority of news coverage is on "bad news", news that is linked to negative emotions, with Zillmann et al. [50] also finding that this "bad news" receives more attention. The findings in the literature that "bad news" is both more common, and attracts more attention, as well as the trend in our data that shows that age is considered limiting more often than in the overall data, leads us to conclude that public discourse in which age is considered limiting for software developers is more commonly reproduced in popular online media.

When we look at the developer types discussed in the public discourse reproduced in the popular online media we analysed, we found that it follows the trend in the overall data, with

stayers being discussed the most often, joiners the second most often, and leavers the least often. An exception to this are forum posts, which, despite being absent from many articles and blog posts, were used quite often in the articles and blog posts in which they appear, discussed joiners more often than any other developer type. Thus, it could be the case that joiners have a larger presence on forums, or that software developers post their experiences on being a joiner to software development on forum posts, forming a niche where leavers are mentioned more often. We also find that demographics, which were again absent from many articles, most often discuss leavers, suggesting that demographics could be used to highlight a possible decline in the number of software developers.

With regards to the reaction to the public discourse, we found that the speaker's reaction to the public discourse differs from the overall trend in the data for the public discourse reproduced, with the emphasis being on purely reproducing public discourse and thus staying neutral, and not reacting to it. This leads us to conclude that the public discourse being reproduced in popular online articles, is mostly used to present information, rather than being used as statements that stimulate a direct reaction.

We thus conclude that the public discourse on age in software development that is reproduced in popular online media presents ageing as limiting in software development, more so than in popular online media as a whole. The focus of the public discourse that is reproduced in popular online media does not differ from the general focus of the popular online media, with the developer type stayers mentioned the most often, followed by joiners, and then leavers. The majority of the time that public discourse on age in software development is used, it is used in a neutral manner, with the speaker reproducing the public discourse.

5.2 Reported Age-Related Aspects

We next discuss our findings for the research question RQ2: "Which age-related aspects are reported in popular online media?", and what implications our findings could have.

We start by discussing the overall trends we found in the data, discussed in Section 4.3.2, for the research question RQ2: "Which age-related aspects are reported in popular online media?".

We found that professional challenges and the career path of software developers are mentioned so often, it could suggest that they are very numerous or considered important issues for the coded articles and blog posts. We found that social challenges and other reported aspects are also well known, being present in around half of the articles. However, they are possibly considered less important or are less numerous than professional challenges or aspects of the career path of software developers, being mentioned less often overall.

When we looked at the count of the developer types, and whether age is considered limiting, for RQ2: "Which age-related aspects are reported in popular online media?", we found that there were no major deviations from the overall trend of the data. As such, stayers are mentioned the most often, and age is considered limiting for the majority of age-related aspects. For the reactions to the public discourse, we found that agreeing was more common that in the overall data, suggesting that the speakers do indeed think that the age-related aspects mentioned in the public discourse are present for software developers.

Some deviations from the overall trend are that for social challenges, we found that they seem to be aimed more at joiners, suggesting they are more of an issue when joining software development later in life. Joiners were also mentioned less often when talking about career path, with leavers being mentioned more often. This suggests more career path related issues occur further in ones career in software development. We also found that the other reported age-related aspects disagree with the public discourse.

We will next discuss the four codes created for the research question RQ2: "Which age-related aspects are reported in popular online media?" in more detail, including the subcodes created for them, followed by a discussion of the employability strategies found.

5.2.1 Social challenges

The social challenges that software developers face according to the public discourse include difficulty fitting in with young people, and difficulty combining work and family obligations.

When we look at fitting in with young people, we found examples of older software developers being in the minority, and feeling out of place. We look at literature on this issue, including in other fields, to see if we can find further examples of older workers not fitting in, and causes for this.

One possibility for older developers having to fit in, is that they are considered less suitable for certain jobs. Finkelstein et al. [15] found in their meta-analysis that older candidates were rated as less suitable for so-called younger types of jobs. In combination with this, we found that being a software developer is a job perceived to be for younger people [9]. Another possibility being from a paper by Kunze et al. [28], which found that increased age diversity was related to an increase in the age discrimination climate. This, somewhat counterintuitively, means that with an increase of age diversity on the company level, the perceived age discrimination climate also increases.

We next look at family obligations, where, in the articles and blog posts, we found examples of clashes occurring between work obligations and family obligations for software developers as one got older. We once again look to literature to investigate this issue from a broader context and compare it to our findings from the articles and blog posts.

We find that clashes between work obligation and family obligations, and in particular a shift towards family obligations, is not only limited to software developers, but is present for engineers as a whole [2]. In fact, conflict can occur any time that work or family obligations make it difficult to fulfil the other [19], no matter the field of work. It thus seems reasonable to assume that this issue is not limited to software developers, but could possibly be worsened by larger demands on one's time due to continuous learning, or as put in article A13: *For all of my other jobs, when my day was over, I left and went home to try and not think about work. Now in tech, I often dont go home but go to meetups straight after work a couple of times a week. At weekends I attend hackathons, I travel to conferences or organize events.*

We thus conclude that, although the social challenges do not seem to be limited to software development, aspects of software development, such as continuous learning and being a software developer being considered a job for a young person, could be exacerbating these issues.

5.2.2 Professional challenges

The professional challenges that software developers face according to the public discourse include that younger developers are cheaper/preferred, pressure to stay relevant and keeping up with new tech, and that experience is considered disadvantageous for software developers.

That younger developers are preferred, could once again be linked to Finkelstein et al. [15] found in their meta-analysis that older candidates were rated as less suitable for so-called younger types of jobs, and that being a software developer is a job perceived to be for younger people [9]. That younger workers are cheaper is something that holds true in most fields. Another issue that is brought up, that there are just fewer senior positions available. This is true for all fields, and is not something that is limited to software development.

Pressure to stay relevant and keeping up with new tech does not seem to be limited to just software developers at first glance. Many fields, such as medicine [22] or engineering [14], require continuous learning. The questions thus becomes, what makes software development different from other professions which require continuous learning. A possible explanation can be found when looking at experience being disadvantageous. The main issue with experience in software development from the articles and blog posts, is that experience is considered irrelevant if it is not with a certain technology that the company uses, as shown in A18: *If you've been working for ten years on a legacy codebase, it doesn't automatically follow that you can leap straight into a programming role on a more modern product.*

Overall, we find that software developers do seem to face a few unique professional challenges

from the professional challenges found in the articles and blog posts. However, the majority do not seem to be limited to software development.

5.2.3 Career path

Aspects related to the career path of software developers mentioned in the public discourse include changing jobs, the declining employability of older developers, and that salary increases stop at a certain age.

When we look at changing jobs, we find that the view of moving to management exists beyond just software development, also existing in the field of engineering as a whole [2]. We also find mention of moving to specialised roles with ones profession, again in engineering as a whole, and not online in software development [2].

We find that declining employability of older developers is also a problem for workers outside of software development [1]. The main difference is that, according to the popular online discourse one could possibly be considered "old" at age 30, whereas the definition of "old" in the study by Geri Adler and Don Hilber [1] is between 55 and 64 years old.

When we look salary increases stopping at a certain age, we in fact find that this is often supported by a book by Clair Brown and Greg Linden [7]. This book talks about engineers in the semiconductor industry, and has already been generalised to include other engineers, and also software developers.

Overall, we again find that the issues faced by software developers that are mentioned in popular online media are also found in other fields. The aspects related to the career path of software developers seem to be related to those in the field of engineering.

5.2.4 Age-related aspects - Other

The other age-related aspects that software developers face according to the public discourse include that older software developers are steady performers and more loyal, that older software developers are more knowledgeable and have more experience, and that older software developers are able to be mentors for younger software developers.

We find that older workers being steady performers and more loyal is not just limited to software developers. A study by Brooke [5] found that older workers, aged 45 and over, had a duration of employment that was 2.4 times higher than workers age 44 and under. We also find that older workers are seen as being as more knowledgeable and more experienced, as well as being in a good position to become mentors for younger workers, in engineering as whole [39].

Thus, we conclude that the other age-related aspects that are mentioned in popular online media, are not specific to software developers. They rather seem to be applicable to older workers as a whole.

5.2.5 Employability strategies

We also look at the employability strategies we found, including that software developers can grow as a software engineer, change their work environment, and try to appear young.

For growing as a software engineer, we look at mastering modern technology, specialising into certain roles, and being a mentor for younger developers. For mastering modern technology and specialising into certain roles, we had also collected responses from Hacker News. In particular, we found that the advice of mastering modern technology was not seen as something that would necessarily be advantageous.

Changing the work environment can involve moving to management, for which we earlier found that moving to management is something that is also considered in engineering as a whole, and not just in software development [2]. From the responses we had collected from Hacker News on the advice of moving to management, we found a positive response, with people seeing the move to management as "safe", although software developers may not be prepared for the move.

When looking at the strategy of appearing young, we found advice such as adapting one's CV to hide graduation dates or removing old jobs. We could not find mention of this in literature, possibly suggesting that this is an issue that has not been researched. Another, rather extreme, aspect found, was the use of plastic surgery to appear younger. Although we could not find any literature on the occurrence of plastic surgery in certain professions, we did find an article [1] from Aurora Clinics [2], a UK based plastic surgery group, where they stated that they had noticed that teachers were the most common profession opting for plastic surgery, although this is not definite proof that plastic surgery is a strategy that is considered in other careers.

Thus, we conclude that, the employability strategies mentioned in the public discourse, we find that, similar to earlier age-related aspects discussed, these aspects were also discussed in broader fields, primarily in engineering as a whole.

5.2.6 Conclusion

Combining the findings and discussion above, we find that for certain age-related aspects, such as pressure to stay relevant and keeping up with new tech, and experience being disadvantageous, we found no mention outside of software development. This could either be because these problems are specific to software development, or that missed literature that discusses these aspects in other fields. In both cases, these age-related aspects could warrant more research if they are negative, both into the actual extent and effect of these aspects, and towards possible solutions.

The majority of the age-related aspects that are found in popular online media however, such as family obligation, changing jobs, and older developers being mentors, are found in wider areas than just software development, with some being applicable to older workers in general. This does not imply that these age-related aspects are not limiting, or that they should not be taken seriously, or even that the extent to which they are present in software development is larger than in other fields.

The fact that these age-related aspects are present in other fields than software development presents both advantages and disadvantages. It is advantageous, in that solutions to problematic age-related aspects could be adapted from other fields to fit software development, resulting in these issues possibly being solved with less effort. However, it could also indicate that these issues permeate society at a deeper level than just certain professions, thus requiring more extensive solutions.

5.3 Reaction to Public Discourse

We will next discuss our findings for the research question RQ3: "What is the reaction to the public discourse in the popular online media?", and what implications our findings could have.

In Section 4.3.3 we found that the use of speculation and personal experiences both seem to be a common way to respond to the public discourse in the articles and blog posts that we coded, with the use of speculation being more common.

We found that the trend of how often the different reactions were used, matched the trend in the overall data. With the majority of reactions to the public discourse perceive age as limiting in software development. For the different developer types we found that stayers are mentioned the most often, but that joiners are mentioned in a higher ratio than in the data as a whole.

We did find that the reactions to the public discourse are different when compared the trend in the overall data. The reaction to the public discourse is much more focused on giving an actual reaction, rather than reproducing other public discourse, with the reactions being mixed when compared to the overall trend in the data. As the reactions to the discourse are so mixed, with reactions disagreeing to the public discourse being slightly more common, it suggests that there is no common consensus in popular online media on the public discourse. This is somewhat confirmed when we look at the reactions we gathered through the forum posts on Hacker News to

[1] https://www.aurora-clinics.co.uk/face/teachers_plastic_surgery/
[2] https://www.aurora-clinics.co.uk/about/

employability strategies. There, we once again found that the reactions were mixed to what was being recommended in the articles and blog posts.

One larger deviation from the trend in the data as a whole that we did find, is that personal experiences seem to mostly be used to talk about joiners, with only few personal experiences mentioning stayers or leavers.

We thus conclude that the reactions to the public discourse in the popular online media present ageing as limiting in software development, although this does not differ significantly from the overall trend in the data. The focus of the reactions to public discourse differs from the general focus of the popular online media slightly, with the developer type stayers mentioned the most often, followed by joiners being mentioned at a higher rate than in the data as a whole, and then leavers. However, when reacting with personal experiences, joiners are the main subject. There is no common consensus in popular online media on whether the public discourse is correct or not. The reactions to the public discourse do differ from the overall trend in the data in that reproduction of other public discourse is much less common.

5.4 Comparison to Literature

We will next compare our findings in popular online media on age(ing) in software development, against the finding in literature on the actual effects of age(ing) in software development, to find areas where discrepancies still exist between perceived and actual effects, as well as to point out different areas of focus between popular online media and literature.

The first difference we discuss is the difference in the definition of "old". We found that the most common definition of "old" in the articles and blog posts was 40 years old and older. When we compare this against the definitions of age used in the literature found in the previous literature review, summarised in Chapter 2, we find that they do not match and that one is considered "old" much earlier in the public discourse. The youngest definition of "old" used in literature is the term "veteran", which is being at least 15 years of experience in software development and who are at least 40 years of age, with others defining "old" as 50 years old and older. This could imply that the literature investigating age(ing) in software development is underestimating when the effects of age(ing) start, or miss certain information by not including developers that are "old" according to the public discourse. A possible reason for the more common definition of "old" being 40 years old and older in the public discourse may be that 40 is the threshold for protection under the US Age Discrimination in Employment Act [3].

In the literature review, we found that literature on older software developers found no decrease in programming ability or found in research until later on in life which can then be mediated by experience [34] [35] [25]. However, this was still mentioned mentioned in the public discourse, in the form of being "stale" or "set in their ways", for example, in article A9: *Fresh-out-of-college kids are appealing employees because they don't know everything yet, but they're open-minded and malleable (although I don't like that word because it has negative connotations). One legitimate fear with older employees is that they're set in their ways.* The increased amount of experienced that older developers have is even considered negative in some of the public discourse, as it may not be considered relevant, as seen in article A18: *And thats exactly what experience does: it looks back. As a result, years of experience in tech isn't always perceived to be as valuable as it is for other industries. Employers (wrongly) assume that experience implies a stagnated skillset, or outdated expectations.* However, in the literature it was found that knowledge of an older technology can improve the knowledge of the successor technology [34] [35] [25].

We found mention of the career path of developers in both literature, as well as the public discourse. Moving to a non programming role was found to be a fundamental part of a software developers career [6] [9]. In the public literature, we also found that software developers move to management as part of their career, such as in article A4: *Other programmers are inevitably promoted to management.* Another aspect mentioned by both literature and public discourse is family obligations, and how family and work obligations can start to conflict later on in life [6] [9].

[3] `https://www.eeoc.gov/age-discrimination`

Something that we do not find any mention of in the public discourse, that is mentioned in literature, is the change in motivation for older developers.

Looking at employability strategies, we found that the advice given in the literature, focuses on what employers can do to improve the current situation, whereas the advice in the public discourse focuses on what developers can do to stay, or find, employment. A common point that both literature and public discourse point at is the adoption of measures to tackle age stereotypes through the composition of teams as well as influencing the age image and corporate culture.

When we look at the publication dates of the articles and blog posts, we found that they were all published in 2009 or later, with the majority being published in 2017 or later. When we compare this to literature that also reports on and gives advice to improve the situation regarding age in software development from the literature review, performed previously and discussed in Chapter 2, we found that many of the articles and blog posts were published later, with many of the papers in the literature review being published in or earlier than 2009. We could thus expect to no longer find certain issues being mentioned in the public discourse published after the literature, if the improvements suggested in the literature has been adopted and had an effect. However, as we find above, certain issues such as the career path of software developers are still being discussed and presented as limited by age.

Overall, we find that the literature on the effects of age(ing) in software development and the public discourse on age(ing) in software development, mostly discuss the same topics, albeit in different contexts in the case of literature, which often uses free/open source software to investigate software developers. However, the literature and public discourse do not always agree, with the literature finding no evidence for a decline in programming ability, and finding that experience with older technology can be beneficial in successor technology. This was not the case in public discourse, in which older developers were found to be "set in their ways" and experience with older tech was often seen not having value. Another major difference between the literature and the public discourse is the difference in the definition of "old", with 40+ years old on average in the public discourse, and the most common definition in the literature being 50+ years old.

5.5 Advice from Literature and Public Discourse

Finally, we will present and shortly discuss advice from literature and the public discourse on how to counter issues on age(ing) in software development.

We first start with advice given in literature found in the literature review summarised in Chapter 2. Suggestions included foresight planning that links industry, labour-market opportunities, and skills development. Further suggestions on how to help older developers stay employed include part time work, flexible hours, contracting back to firms, and home based 'E-work' [6] [9]. It was also pointed out that human resource management should investigate and put into effect measures to tackle age stereotypes through the composition of teams as well as influencing the age image and corporate culture through communications [9] [24].

For the advice from the public discourse, we look to the employability strategies that were found for both individual developers and companies.

For individual software developers, how to cope with possible ageism or challenges in employment depends on personal preferences. For some software developers, changing the work environment or adapting expectations may not be as welcome as for others. We find that the items included in growing as a software engineer, although once again not suitable for everyone, provide possible good recommendations, including diversifying skills, including soft skills, and specialising.

For companies, trying to mitigate the negative aspects found for employability strategies seem to be a good starting place. This includes looking at their hiring process, to be more inclusive in both the language used in actual job advertisements, as well as adapting their interview process to be inclusive towards older developers. Another aspect was changing the view that companies have of experience, where it is currently sometimes viewed as "technology baggage". Changing this view on experience, coincides with our finding in the differences between literature and public

discourse, where experience with older technology, can translate to more modern skills, whereas it is seen as baggage in the public discourse.

Chapter 6

Limitations and Threats to Validity

6.1 Limitations and Validity

In this chapter we discuss limitations of this paper as well as threats to the validity of the results found and conclusions made.

6.2 Limitations

The main limitation of our study, was the lack of experience with the research methods used for the main researcher. This caused certain parts of the research to require much more time than would otherwise be needed.

The overall limited amount of time available to perform the research was another limitation. Coding is recognised to be a time consuming process, and due to the lack of experience with coding of the main researcher, required more time to complete. Furthermore, certain aspects of the research performed, have only been performed by a single researcher, possibly limiting the insights made from the data.

6.3 Construct Validity

We may be missing influential or controversial articles and blog posts in our study due to the inclusion and exclusion being too strict. We could also have missed articles and blog posts due to the selection of keywords introducing bias towards certain articles and blog posts, with the terminology being used in other influential or controversial articles and blog posts differing enough to not be found with our search string.

Furthermore, it could be the case that the items we code for during the coding of the articles and blog posts, give a biased view of the public discourse by design. We have tried to mitigate this, by using open coding, in which the codes are derived from the data.

6.4 Internal Validity

A threat to the internal validity of our study is that certain aspects of the study has been conducted by a single person, rather that multiple people. This is an issue in the subcoding of the results for RQ2 “Which age-related aspects are reported in popular online media?”.

Another threat to the internal validity of our study, is the counting of the number of occurrences of the codes and drawing conclusions from the counts. This may give an incorrect view of the

public discourse, due to longer articles mentioning certain aspects just as a consequence of being longer, and not as the issue mentioned is more important. We have tried to mitigate this by also looking at the amount of articles a code occurs in, when drawing conclusions.

6.5 External Validity

The main external threat we have in this study is that the articles and blog posts we have found and analysed form but a part of the discourse. In particular, we have focused on popular online media from the USA, and thus public discourse from the USA. As such, it may not be the case that the results we found are valid for the discourse on software developers every where in the world, due to discourse differing per region.

Chapter 7

Conclusions

In this thesis we investigated how age(ing) in software development is seen in the public discourse through the four research questions:

RQ1 How is the public discourse on age in software development reproduced in popular online media?
RQ2 Which age-related aspects are reported in popular online media?
RQ3 What is the reaction to the public discourse in the popular online media?
RQ4 How does the public discourse on age(ing) in software development compare to results on the effects of age(ing) in software development found in literature?

Combining our findings for all of the research questions, we found that age(ing) in software development is mainly presented as limiting, especially when being reproduced, with people staying in software development being the main focus for the public discourse. Age-related aspects that are mentioned include social challenges, such as work and family obligations clashing, professional challenges, such as pressure to keep up with new tech, and career path aspects such as changing jobs. We did not find a common consensus on whether age is actually limiting with reactions being split between agreeing, disagreeing, and purely reproducing other public discourse. The majority of the age-related aspects discussed in the public discourse are also be found in literature. However, some of these findings run counter to findings in the literature, in which age is not seen as limiting with regards to technical ability.

7.1 Future Research

The public discourse studied in this thesis is just a small part of the discourse on age(ing) in software development. Further studies could focus on different types of media in which discourse on age(ing) in software development is found, to examine whether similar aspects are being mentioned.

Furthermore, we have only included media from the USA. Future research can investigate how the discourse on age(ing) in software development differs, and what aspects seem to be common, between different countries and cultures. This can help countries individually assess which issues are most pressing, and counter negative aspects surrounding age(ing) in software development found in the discourse that are presented to be limiting software developers.

In our research we found that there exists a difference between the definition of “old” used in the public discourse we found, and the definition of “old” used in literature found in the previous literature review. This could cause discrepancies between findings in literature, and what is being said and experienced. This could also limit the effectiveness of measures meant to counter ageism proposed in literature. As such, future research could investigate the aspects mentioned in the public discourse and literature on age(ing) in software development using the definition of the age from the public discourse. It could also verify that the proposed measures meant to counter

ageism that have been adopted, have not left out certain people due to the different definition of "old".

Bibliography

[1] Geri Adler and Don Hilber. Industry hiring patterns of older workers. *Research on Aging*, 31(1):69–88, 2009. 47

[2] Lotte Bailyn and John T. Lynch. Engineering as a life-long career: Its meaning, its satisfactions, its difficulties. *Journal of Occupational Behaviour*, 4(4):263–283, 1983. 46, 47

[3] B. Berelson. *Content analysis in communication research*. Free Press, 1952. 7

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. OReilly Media, Inc., 1st edition, 2009. 14

[5] Libby Brooke. Human resource costs and benefits of maintaining a mature-age workforce. *International Journal of Manpower*, 24:260–283, 05 2003. 47

[6] Libby Brooke. Prolonging the careers of older information technology workers: continuity, exit or retirement transitions? *Ageing and Society*, 29(2):237256, 2009. 1, 5, 6, 49, 50

[7] Clair Brown and Greg Linden. *Chips and Change: How Crisis Reshapes the Semiconductor Industry*. The MIT Press, 2009. 47

[8] Carol Busch, Paul S. De Maret, Teresa Flynn, Rachel Kellum, Sheri Le, Brad Meyers, Matt Saunders, Robert White, and Mike Palmquist. Content analysis. Writing@CSU. Colorado State University. Available at `https://writing.colostate.edu/guides/guide.cfm?guideid=61`, 1994-2012. 7

[9] Tammy Duerden Comeau and Candace L. Kemp. Intersections of age and masculinities in the information technology industry. *Ageing and Society*, 27(2):215232, 2007. 5, 6, 46, 49, 50

[10] J. L. Davidson, R. Naik, U. A. Mannan, A. Azarbakht, and C. Jensen. On older adults in free/open source software: reflections of contributors and community leaders. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, July 2014. 5

[11] Jennifer L. Davidson, Umme Ayda Mannan, Rithika Naik, Ishneet Dua, and Carlos Jensen. Older adults and free/open source software: A diary study of first-time contributors. In *Proceedings of The International Symposium on Open Collaboration*, OpenSym '14, pages 5:1–5:10. ACM, 2014. 5

[12] Joanna F. DeFranco and Phillip A. Laplante. A content analysis process for qualitative software engineering research. *Innovations in Systems and Software Engineering*, 13:129-141, 2017. vi, 7, 8

[13] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. *Selecting Empirical Methods for Software Engineering Research*, pages 285–311. Springer London, London, 2008. 7, 9

[14] Clive Ferguson. The continuous professional development of engineers and flexible learning strategies. *International Journal of Lifelong Education*, 17(3):173–183, 1998. 46

[15] Lisa Finkelstein, Michael Burke, and Manbury Raju. Age discrimination in simulated employment contexts: An integrative analysis. *Journal of Applied Psychology*, 80:652–663, 12 1995. 46

[16] Joseph L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378-382, 1971. 12

[17] Najlah Gali, Radu Mariescu-Istodor, and Damien Hostettler. Framework for syntactic string similarity measures. *Expert Systems with Applications*, 129, 04 2019. 14

[18] G. Gibbs. *Analyzing qualitative data*. Sage qualitative research kit. Sage, 2007. 10, 12

[19] Jeffrey H. Greenhaus and Nicholas J. Beutell. Sources of conflict between work and family roles. *Academy of Management Review*, 10(1):76–88, 1985. 46

[20] Jack B. Haskins, M. Mark Miller, and Jan Quarks. Reliability of the news direction scale for analysis of the good-bad news dimension. *Journalism Quarterly*, 61(3):524–528, 1984. 44

[21] Andrew F. Hayes and Klaus Krippendorff. Answering the call for a standard reliability measure for coding data. *Communication Methods and Measures*, 1(1):77-89, 2007. 12

[22] Hans Asbjørn Holm. Quality issues in continuing medical education. *BMJ*, 316(7131):621–624, 1998. 46

[23] Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. Words versus character n-grams for anti-spam filtering. *International Journal on Artificial Intelligence Tools*, 16:1047-1067, 12 2007. 14

[24] Heidrun Kleefeld. *Demografischer Wandel und Kompetenz zur Innovation in der IT-Branche — Anforderungen an ein strategisches Human Resource Management*, pages 101–127. Gabler, 2008. 6, 50

[25] Grzegorz Kowalik and Radoslaw Nielek. Senior programmers: Characteristics of elderly users from stack overflow. In *Social Informatics*, pages 87–96. Springer International Publishing, 2016. 5, 49

[26] Ralf T. Krampe and Neil Charness. *Aging and Expertise*, pages 835–856. Cambridge University Press, 2 edition, 2018. 5

[27] K. Krippendorff. *Content Analysis: An Introduction to Its Methodology*. SAGE Publications, 2018. 7

[28] Florian Kunze, Stephan A. Boehm, and Heike Bruch. Age diversity, age discrimination climate and performance consequencesa cross organizational study. *Journal of Organizational Behavior*, 32(2):264–290, 2011. 46

[29] I. Lopez-Gazpio, Montse Maritxalar, M. Lapata, and Eneko Agirre. Word n-gram attention models for sentence similarity and inference. *Expert Systems with Applications*, 132:1-11, 10 2019. 14

[30] Uwe Lünstroth. *Demography, Aging and High Tech - The Case of Software Developers*. Lit Verlag, 2006. 6

[31] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. 13, 14

[32] Victor W. Marshall. A life course perspective on information technology work. *Journal of Applied Gerontology*, 30(2):185–198, 2011. 5

[33] Tim Menzies, Laurie Williams, and Thomas Zimmermann. *Perspectives on Data Science for Software Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2016. 10

[34] P. Morrison and E. Murphy-Hill. Is programming knowledge related to age? an exploration of stack overflow. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 69–72, May 2013. 5, 49

[35] P. Morrison, R. Pandita, E. Murphy-Hill, and A. McLaughlin. Veteran developers' contributions and motivations: An open source perspective. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 171–179, Sep. 2016. 5, 49

[36] Sik Hung. Ng and James J. Bradac. *Power in language : Verbal communication and social influence*. Sage Publications Newbury Park, 1993. 2

[37] Sik Hung. Ng and Fei Deng. Language and power, 08 2017. 2

[38] CBS - werkzame beroepsbevolking; beroep. `http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=82808ned&D1=0&D2=0&D3=7-14&D4=120-121,123-124&D5=4,9,14,19,24,29,34,39,44,49,54,59,64,69,74&HDR=G4,T,G1&STB=G2,G3&VW=T`. Accessed: 2019-01-05. 1

[39] Michelle L. Oppert and Valerie OKeeffe. The future of the ageing workforce in engineering: relics or resources? *Australian Journal of Multi-Disciplinary Engineering*, 15(1):100–111, 2019. 47

[40] Richard A. Posthuma and Michael A. Campion. Age stereotypes in the workplace: Common stereotypes, moderators, and future research directions. *Journal of Management*, 35(1):158–188, 2009. 5

[41] Jing Quan, Ronald Dattero, and Stuart D Galup. Impact of age on information technology salaries. In *Global, Social, and Organizational Implications of Emerging Information Resources Management: Concepts and Applications*, pages 403–420. IGI Global, 2010. 5

[42] Johnny Saldaña. *The Coding manual for Qualitative Researchers*. SAGE Publications Ltd, 2 edition, 2013. 10, 11, 12

[43] Pavel Stefanovič, Olga Kurasova, and Rokas Štrimaitis. The n-grams based text similarity detection approach using self-organizing maps and similarity measures. *Applied Sciences*, 9, 05 2019. 14

[44] Sami Surakka. Analysis of technical skills in job advertisements targeted at software developers. *Informatics in Education*, 4:101-122, 2005. 7

[45] Bureau of Labor Statistics - labor force statistics from the current population survey - 2011. `https://www.bls.gov/cps/aa2011/cpsaat11b.htm`. Accessed: 2019-01-05. 1

[46] Bureau of Labor Statistics - labor force statistics from the current population survey - 2017. `https://www.bls.gov/cps/cpsaat11b.htm`. Accessed: 2019-01-05. 1

[47] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 38:1–38:10. ACM, 2014. 4

[48] United Nations - Ageing. `http://www.un.org/en/sections/issues-depth/ageing/`. Accessed: 2018-12-28. ii, 1

[49] Aiwu Xia and Brian H. Kleiner. Discrimination in the computer industry. *Equal Opportunities International*, 20(5/6/7):117–120, 2001. 5

[50] Dolf Zillmann, Lei Chen, Silvia Knobloch, and Coy Callison. Effects of lead framing on selective exposure to internet news reports. *Communication Research*, 31(1):58–81, 2004. 44

[51] Thomas Zimmermann. Card-sorting: From text to themes. Available at `https://github.com/ds4se/chapters/blob/master/zimmermann/card-sorting.md`, 2 2016. 13

Appendix A

Literature Study

Effects of ageing for software developers in the work environment

George Park	g.w.a.park@student.tue.nl	0852572
Alexander Serebrenik	a.serebrenik@tue.nl	Supervisor
Sebastian Baltes	sebastian.baltes@adelaide.edu.au	Supervisor

May 16, 2020

Abstract

The western world has an increasing amount of older software developers, however, not much is known about challenges that these developers face and how they are perceived as a group. In this paper we conduct a literature review of 18 pieces of literature to investigate our research question: "What are the effects of ageing for software developers in the work environment?". We find that older developers are motivated by different factors, are perceived to be different from, and do not perform less well when compared to younger developers. We find that the main limitation of the current literature is the lack of a common definition of 'older'. We also present possible areas for future research, such as performing longitudinal studies, investigating which measures are most effective to prevent or decrease age stereotypes, and investigating age stereotypes in different cultures.

1 Introduction

As the population continues to age [26], progressively more countries across the world adopt policies aimed at keeping older works in the workforce. This also has the implication that there is, and will continue to be, a larger amount of older software developers. In the Netherlands, the number of software developers between 55 and 65 increased from 6000, 4.3% of all developers, in 2003 to 26000, 12% of all developers, in 2017 [17]. In Germany, there was a total of 301,000 software developers as of 2016, of which 68,000, 23% of all developers, are between 45 and 55 years old and 33,000, 11% of all developers, are between 55 and 65 years old [6]. In the USA, the number of software developers of 55 to 64 years old increased from 87,000, 8.3% of all developers, in 2011 [23], to 175,000, 11% of all developers, in 2017 [24]. However, relatively little is known about what the effects of ageing are for software developers, and for software development as a whole.

Understanding the challenges facing older software developers, and knowing more about their needs and what keeps them motivated, can be beneficial to both businesses and software developers themselves. It can help businesses keep older employees productive, as well as help fill the need for developers by preventing skilled workers from leaving.

This leads us to the following research question: "What are the effects of ageing for software developers in the work environment?". We investigate the research question through a formal literature review. Besides specifically investigating the effects of ageing for software

developers in the work environment, we also look at the perception that software developers have on ageing and older software developers, and the effects of ageing in general.

The remainder of the paper is outlined as follows. In Section 2 we present the methodology of the the literature review we conducted and the steps we made and choices we made. In Section 3 we present some effects of ageing in general, and in Section 4 we present the effects of ageing for software developers in the work environment that have been found in the literature. In Section 5 we present our findings on limitations of the current research and possible areas for future research. In Section 6 we discuss limitations of our literature study and in Section 7 we give a summary of the paper.

2 Method

To investigate the research question, we proceed with a literature study. For the basic planning and motivation of the research, the guidelines for systematic literature studies by Kitchenham [8] were used. For the actual literature review, we use the snowballing procedure as presented by Wohlin [25].

2.1 Inclusion and Exclusion Criteria

We have the following inclusion and exclusion criteria for literature in our literature review. We include peer reviewed literature published before 2019, as well as non peer reviewed literature in the form of books or chapters from books published before 2019, on the following topics:

- Age stereotypes in IT
- Perception of older software developers
- Effects of ageing on software developers

We remove duplicate studies, with the most complete version of the study being included in the literature review.

We decide on a piece of literature being using the following process. We first look at the title to see if it tentatively literature to include. If it is still unsure whether it should be included of excluded, we continue by reading the following items: the abstract, introduction, section and sub-section headings, and the conclusions. After reading these, or after reading only a subset of the items, we decide if we include of exclude the piece of literature.

Due to the literature study being undertaken by a single person, the final decision whether to include or exclude literature is also made by a single person.

2.2 Data Collection

We extract the following data from the literature that we include in the review:

- Source and full reference
- Authors
- Publication year
- Publication venue
- Language
- Title (Original and translated)

- Summary of the study including the main research questions and the findings
- Limitations of the study as reported by the authors
- Future research directions as identified by the authors

We again have that due to the literature study being undertaken by a single person, the data extraction was also done by a single person. The detailed data collection can be found in Appendix A, Appendix B, and Appendix C.

2.3 Start Set

The first step that needs to be taken when using the snowballing approach is the identification of a starting set of literature for the snowballing procedure, which we obtain through a database search using an appropriate search string. We have decided to use two search strings for our initial search, namely "perception ageing software developers" and "perception ageism software development". We have chosen to include perception in both search strings as we also want to find results on how software developers perceive ageing. We also include "perception ageism software development" to find results on common ageist perceptions that are present in software development.

As the chosen search strings are in English, the literature found in the initial search will also be written in English. This may be partially compensated for by the snowballing process.

To avoid publisher bias, we use Google Scholar to search for a starting set of literature using the search string "perception ageing software developers" as well as the string "perception ageism software development". In total, we found 17,100 hits for the first search string and 7.820 hits for the second search string when not including patents and citations. Due to the limited amount of time we have to conduct this research, we limit ourselves to the first 100 results of each search string as the relevance of the search results were found to drop off after the first 100 results. We take the union of these results, giving a total of 200 results, on which we apply the inclusion and exclusion criteria as described in Section 2.1.

In total 10 candidates were found that passed the inclusion and exclusion criteria, with each piece of literature being denoted L1, L2 to indicate that they are literature that have been included. The literature L1 up to and including L10, found in Section 2.4.4, were thus included as the starting set.

2.4 Iterations

After a starting set of literature had been identified, we continue with iterations in which we conduct backward and forward snowballing. We repeat this process on any new literature found in the previous iteration until no new relevant literature is found. Note that for deciding whether or not to include or exclude literature in backward and forward snowballing, we also look at how it is referenced. We do this after looking at the title, and before continuing with the abstract.

2.4.1 First Iteration

Backward Snowballing

We start with backward snowballing, in which we look at the references of the included literature that we have not yet completed yet. As we are in the first iteration of snowballing, the included literature that we have not yet completed yet is the starting set of literature.

We find 76 references for L4, 46 references for L5, 50 references for L6. From L4, L5 and L6 we each find one new relevant reference, which we call L11, L12 and L13. We find 85 references for L7, of which we find two new relevant references, which we call L14 and L15. The literature L11 up to and including L15, found in Section 2.4.4, were thus included and added to the complete literature set.

We find that L1 has 38 references, of which one could possibly be relevant for us. J. McMullin, (2004). Ageism in Information Technology Employment. However, we could not find any kind of record for it and thus find no relevant references.

Furthermore, L2 has 39 references, L3 has 41 references, L8 has 36 references, L9 has 120 references, and L10 has 89 references. However, for L2, L3, L8, L9, L10 we do not find any new relevant literature as all referenced literature is excluded by the inclusion and exclusion criteria after reading either the title or abstract or has already been included.

Forward Snowballing

We then continue with forward snowballing, in which we find citations of the included literature that we have not yet completed yet. As we are in the first iteration of snowballing, the included literature that we have not yet completed yet is the starting set of literature. We look for citations in all iterations by using Google Scholar.

We find that L1 is cited 31 times and we find one citation which is relevant and which we call L16. The literature L16, found in Section 2.4.4, was thus included and added to the complete literature set.

L2 is cited 27 times, L3 is cited 5 times, L4 is cited 5 times, L5 is cited 0 times, L6 is cited 8 times, L7 is cited 0 times, L8 is cited 11 times, L9 is cited 2 times, L10 is cited 1 time. However, for L2 through L10 we do not find any new relevant literature as all citing literature is excluded by the inclusion and exclusion criteria after reading either the title or abstract or has already been included.

2.4.2 Second Iteration

Backward Snowballing As we are in the second iteration of snowballing, the included literature that we have not yet completed yet is the set of literature found in the first iteration of snowballing.

We find that L14 has 15 references and L16 has 23 references. We find find one reference to new relevant literature in both L14 and L16. However, we can not find the literature mentioned in L14 anywhere, U. Lünstroth, (2001), Der älter werdende Software-Experte. We also have that we can not find an online version of the literature mentioned in L16 anywhere, J.A. McMullin, & T.D. Comeau, (2011). Aging and Age Discrimination in IT firms, in Age, Gender, and Work: Small Information Technology Firms in the New Economy, edited by J.A. McMullin. Kelowna: UBC Press. It would have been possible to order a hard copy of the book, however, this was decided against due to the time restraints.

Forward Snowballing We again have that the included literature that we have not yet completed yet is the set of literature found in the first iteration of snowballing.

L15 is cited 52 times, and we find two citations by new relevant literature which we name L17 and L18 respectively. The literature L17 and L18, found in Section 2.4.4, were thus included and added to the complete literature set.

L11 is cited 12 times, L12 is cited 10 times, L13 is cited 14 times. However, for L11, L12, and L13 we do not find any new relevant literature as all citing literature is excluded

by the inclusion and exclusion criteria after reading either the title or abstract or has already been included. L14 and L16 could not be found on Google Scholar and we thus also find no references for them.

2.4.3 Third Iteration

Backward Snowballing As we are in the third iteration of snowballing, the included literature that we have not yet completed yet is the set of literature found in the second iteration of snowballing.

L17 has 18 references, and L18 has 118 references. However, for both L17 and L18 we do not find any new relevant literature in the referenced literature.

Forward Snowballing We again have that the included literature that we have not yet completed yet is the set of literature found in the second iteration of snowballing.

L17 is cited 12 times, and L18 is cited 10 times. However, for both L17 and L18 we do not find any new relevant literature in the citing literature.

2.4.4 Complete literature set

- L1: T. D. Comeau, & C. L. Kemp, (2007). Intersections of age and masculinities in the information technology industry. Ageing & Society, 27(2), 215-232. [3]
- L2: L. Brooke, (2009). Prolonging the careers of older information technology workers: continuity, exit or retirement transitions?. Ageing & Society, 29(2), 237-256. [2]
- L3: J. L. Davidson, R. Naik, U. A. Mannan, A. Azarbakht, & C. Jensen, (2014, July). On older adults in free/open source software: reflections of contributors and community leaders. In Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on (pp. 93-100). IEEE. [5]
- L4: U. Schloegel, S. Stegmann, A. Maedche, & R. van Dick, (2018). Age stereotypes in agile software developmentan empirical study of performance expectations. Information Technology & People, 31(1), 41-62. [21]
- L5: J. Quan, R. Dattero, & S. D. Galup, (2010). Impact of Age on Information Technology Salaries. In Global, Social, and Organizational Implications of Emerging Information Resources Management: Concepts and Applications (pp. 403-420). IGI Global. [19]
- L6: U. Schloegel, S. Stegmann, A. Maedche, & R. van Dick. (2016). Reducing age stereotypes in software development: The effects of awareness-and cooperation-based diversity interventions. Journal of Systems and Software, 121, 1-15. [20]
- L7: P. Morrison, R. Pandita, E. Murphy-Hill, & A. McLaughlin, (2016, September). Veteran developers' contributions and motivations: An open source perspective. In Visual Languages and Human-Centric Computing (VL/HCC), 2016 IEEE Symposium on (pp. 171-179). IEEE. [16]
- L8: J. L. Davidson, U. A. Mannan, R. Naik, I. Dua, & C. Jensen, (2014, August). Older adults and free/open source software: A diary study of first-time contributors. In Proceedings of The International Symposium on Open Collaboration (p. 5). ACM. [4]
- L9: U. Schloegel, S. Stegmann, R. van Dick, & A. Maedche (2018). Age stereotypes in distributed software development: The impact of culture on age-related performance expectations. Information and Software Technology, 97, 146-162. [22]

- L10: N. Kock, M. Moqbel, Y. Jung, & T. Syn (2018). Do older programmers perform as well as young ones? Exploring the intermediate effects of stress and programming experience. Cognition, Technology & Work, 1-16. [10]
- L11: V. W. Marshall, (2011). A Life Course Perspective on Information Technology Work. Journal of Applied Gerontology, 30(2), 185198. [14]
- L12: A. Xia, B. H. Kleiner, (2001) "Discrimination in the computer industry", Equal Opportunities International, Vol. 20 Issue: 5/6/7, pp.117-120. [27]
- L13: H. Kleefeld (2008) Demografischer Wandel und Kompetenz zur Innovation in der IT-Branche Anforderungen an ein strategisches Human Resource Management. Eisenkopf A., Opitz C., Proff H. (eds) Strategisches Kompetenz-Management in der Betriebswirtschaftslehre. Gabler. [9]
- L14: U. Lünstroth, (2007), Demography, Aging, and High-Tech - The Case of Software Developers. Shaping Better Technologies. Lit Verlag [13]
- L15: P. Morrison and E. Murphy-Hill, (2013), Is programming knowledge related to age? in Companion to the Working Conference on Mining Software Repositories, 14. [15]
- L16: E. Jovic, & J. McMullin, (2016). Risky Business: Ageing as an Information Technology Worker. Global Ageing in the Twenty-First Century: Challenges, Opportunities and Implications, 195. [7]
- L17: G. Kowalik, & R. Nielek, (2016, November). Senior programmers: Characteristics of elderly users from stack overflow. In International Conference on Social Informatics (pp. 87-96). Springer, Cham. [11]
- L18: S. Baltes, & S. Diehl, (2018, October). Towards a theory of software development expertise. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 187-200). ACM. [1]

3 Ageing in General

In this section, we look at the effects of ageing on aspects that could play a role in the perception software developers. We also look at findings in the literature on age stereotypes in the workplace.

3.1 Ageing and Expertise

Ralf Th. Krampe and Neil Charness [12] review age-comparative studies in different domains in their study on ageing and expertise. They discuss the effects of ageing for older experts as well as what older experts do to maintain their expertise which both could be of interest in our study.

Krampe and Charness find that the results from psychometric tests on general cognitive abilities do not correlate well with expert performance in older age. In other words, although the results from psychometric test on general cognitive abilities may become worse as experts age, this does not mean that their performance in their field of expertise will also become worse.

Overall, Krampe and Charness find that high levels of skill can be maintained by older adults through deliberate practice, and that this is not only the case in knowledge-based skills, but also with skills that involve speed and accuracy. They also note that as older

workers are typically in more stable job positions than younger workers, older workers have an environment in which they can selectively maintain certain skills. This, however, is not the case in jobs for which the demands change over time, and older workers thus risk their skills becoming obsolete.

3.2 Age Stereotypes in the Workplace

Posthuma and Campion [18] conducted a literature review that was published in 2009 on age stereotypes in the workplace. In their study they categorised the stereotypes they found as well as suggesting directions for future research. Results that are of interest to us, are that Posthuma and Campion found stereotypes that older workers have a decreased ability to learn, have less ability, and are less motivated and productive than younger workers.

Not all age stereotypes that Posthuma and Campion found were negative though. They also found that older workers were found to be "more stable, dependable, honest, trustworthy, loyal, committed to the job, and less likely to miss work or turnover quickly" [18].

They also found evidence that certain stereotypes were false. Notably, it was found that work performance did not decrease with age and that individual skill and health are more important to work performance.

Posthuma and Campion noted that the age stereotypes that were found, are particularly strong in certain industries, including IT.

Posthuma and Campion likewise suggested future directions for research. These include how different national cultures influence age stereotypes, how managerial practices can influence age stereotypes and how they can create a environment that reduces the potential for age stereotypes, and if and how training can be employed to reduce age stereotypes. Their suggestions were not limited to research directions however, as they also noted the need for longitudinal studies, as most research is based on cross-sectional design, and that future research has to be careful to avoid socially desirable responding.

4 Ageing in Software Development

In this section, we look at effects of ageing for software developers in the work environment. We first look at the findings from the literature and common findings across studies, after which we look at recommendations that have been made in the literature to combat the age stereotypes or challenges faced by older developers.

4.1 Findings in Literature

4.1.1 Motivation

We will first look at what motivates older developers and how these are different when compared to younger developers.

First, we are interested in whether older developers actually have different motivations when compared to younger developers. In the context of free/open source software, the top motivations found for older developers were intrinsic motivation, community identification, internal values, and altruism. This differs from younger developers who cite career benefits and learning a major motivating factors. Davidson, Mannan, Naik, Azarbakht, and Jensen [5] and Davidson, Mannan, Naik, Dua, and Jensen [4] discuss this in more detail. It was also

found that older developers are biased more toward sense of responsibility by Kowalik and Nielek [11].

We next look at what causes these changes in motivation. The most prominent cause found was responsibilities, such as family needs, becoming more important to older developers with the needs of the team no longer being the most important. The demands placed on developers, such as working overtime and continuous learning, come into conflict with responsibilities, such as family needs, later on in life as found by Comeau and Kemp [3], and Brooke [2].

4.1.2 Perception

We will next look at what the perception is of older developers by other developers and the organisation as a whole as well as how they perceive themselves.

We start by looking at whether older developers are perceived to be different. Age stereotypes we found to affect older developers the most, and that younger developers hold the strongest age stereotypes. It was also noted that the performance expectations of developers declines with age. Furthermore, the effects of age stereotypes in different cultures was studied with age stereotypes toward older employees being the strongest in China and, Poland and Bulgaria out of the four countries studied. Schloegel, Stegmann, van Dick, and Maedche [21], [22] present these finding in more detail.

Next, we look at specific examples of how older adults are perceived by other developers and themselves. Developers were found to think that being able to keep up to date and to be able to continue learning was linked to youth. Thus, older developers would be less adaptive to change. IT workers also have another view on who are considered seasoned or veterans, rather than judging by a persons age, it depends on how long they had worked in the field. Furthermore, in the career path of developers, moving to a non programming role was seen as a fundamental part of ones career. Expressing a wish to continue as a developer was seen as having a lack of drive. Older developers were also seen as more expensive, not only due to higher pay, but also due to medical support. These perceptions are found by Comeau and Kemp [3], Xia and Kleiner [27], Marshall [14].

However, older developers were also perceived to have benefits when compared to younger developers. Marshall [14] found that experience from being in IT for a longer time was seen as an advantage. In the context of free/open source software, Davidson, Mannan, Naik, Azarbakht, and Jensen [5] found the following ten unique benefits that older adults offer:

- Software development experience
- Understanding technology trends
- Life experience as a user, parent, spouse
- Experience in general
- Maturity
- Understanding computer/software architecture
- They may have more time
- They may have more connections/networking
- Wisdom
- Professional experience in general

4.1.3 Performance

Lastly, we will look at what the literature has found on the actual performance of older developers and challenges they face.

Older developers did notice a decline programming ability by themselves, Baltes and Diehl [1], this however does not mean that older developers are worse as a whole. In fact, other studies could not confirm that older developers perform worse than younger developers as a whole. It was found that although age is positively associated with perceived stress, and thus negatively with programming performance, age is also positively associated with programming experience, and thus positively with programming performance. Though the relationship between perceived stress and programming performance decreased as programming experience increased, suggesting that that the perception that older programmers do not perform as well as young ones is false.

In the context of open source, the literature in our study found that, on Stackoverflow, older developers have higher reputation when compared to younger developers, and that older developers could have a broader knowledge that younger developers. This could indicate that programming knowledge can be maintained at a high level into older age. It was also noted that older developers learn new technologies, and that knowledge of a technology can improve the knowledge of the successor of that technology in certain cases which could aid older developers learning new technologies. These findings are presented in more detail by Morrison and Murphy-Hill [15], Morrison, Pandita, Murphy-Hill, and McLaughlin [16], and Kowalik and Nielek [11].

Although we found that older developers do not perform worse than younger developers in the literature, Quan, Dattero, and Galup [19] found that when looking at wages in IT, and when controlling for education and technical experience levels, employees aged 40 or over make only 88% of their younger counterparts. This difference changes however depending on the sector, as well as depending on job function.

4.2 Recommendations in Literature

The literature included in our paper also contained recommendations on how the situation can be improved for older workers. As it was found that employment in IT was often project based, with employees only working at a company for a few projects, opportunities for workers to avoid 'gaps' in employment should be provided. This could be through certain human resources management policies such as providing foresight planning that links industry, labour-market opportunities, and skills development. It could also be provided though part time work, flexible hours, contracting back to firms, and home based 'E-work'. Brooke [2] and Comeau and Kemp [3] discuss these measures in more detail. Lünstroth [13] also discusses these recommendations using case studies illustrating situations in companies where these measure could have supported older workers, or had already supported older workers.

Age stereotypes that these workers face should also be tackled at all levels, as both Brooke [2] and Comeau and Kemp [3] note. In particular, Comeau and Kemp find that the IT industry may disadvantage older workers at the organisational level through the age differentiation of roles. Kleefeld [9] mentions that Human Resource Management should investigate and put into effect measures to tackle these age stereotypes. These measures include influencing the composition of teams as well as influencing the age image and corporate culture through communications. For example, awareness-based and cooperation-based interventions could

be conducted. Schloegel, Stegmann, van Dick, and Maedche [20] discuss an awareness-based workshop and a cooperation- based workshop they conducted in their research.

5 Discussion

In this section, we look at our findings from the literature review to identify gaps in current knowledge and weaknesses in literature. We also point out potential areas for future research and why further research into these areas is important.

5.1 Limitations of Current Literature

We will first note weaknesses in the current research. The main weakness we encounter is the lack of common definition of what researchers consider 'older' and the use of the term 'veteran'. For example, Schloegel, Stegmann, van Dick, and Maedche [20][22][21] use > 50 years old to define older, and Morrison, Pandita, Murphy-Hill, and McLaughlin [16] use the term veteran, which means at least 15 years of experience in software development and who are at least 40 years of age. Davidson, Mannan, Naik, Azarbakht, and Jensen [5] define older as 50 years old or older which which differs only slightly compared with to Schloegel, Stegmann, van Dick, and Maedche. This makes it difficult to be able to draw detailed conclusions from the available literature, and to synthesise the literature, as certain effects or challenges may only start around a certain age which could potentially cause studies to draw different conclusions.

5.2 Future Research

We next discuss some possible areas for future research and why these areas are important to study. A mayor shortcoming in current research is the lack of longitudinal studies, as most research is based on cross-sectional design. The need for these studies has also been mentioned by Posthuma and Campion [18]. Most literature used in this literature review has relied on data from open source or data collected from surveys. Longitudinal studies will be able to gather data over a longer period of time and will be of particular use in studying the effectiveness of methods to decrease negative age stereotypes.

In Section 4.2 we found that the literature studied in this review provided certain recommendations as to how the current situation could be improved. However, not many specific measures, such as training's or workshops, that could prevent and decrease age stereotypes were mentioned. Future research could investigate how certain measures affect age stereotypes and which measures are most effective such that age stereotypes can be tackled more effectively.

As mentioned above, in Section 4.2 we found that the literature studied in this review provided certain recommendations as to how the current situation could be improved. As many of the papers in which these recommendations have been found, Brooke [2], Comeau and Kemp [3], Kleefeld [9], and Lünstroth [13], were published in or earlier than 2009. Future research could investigate whether these recommendations have been adopted and if they have been adopted, if they have had an effect. If these recommendations have not been adopted, then they could be recommended again. If these recommendations have been adopted and they have had no or little effect, then new recommendations should be found.

Lastly, it would also be interesting to conduct research to investigate whether, and to what degree, culture has on age stereotypes in IT. Schloegel, Stegmann, van Dick, and Maedche

[22] already conducted a study on the effects of culture on age stereotypes which could be verified by other studies. Other studies could also investigate other aspects, such as whether different measures are better at preventing an decreasing age stereotypes in different cultures so that the most effective measures can be selected.

6 Limitations and Threats to Validity

In this sections, we discuss limitations of this paper as well as threats to the validity of the results found and conclusions made.

6.1 Limitations of Current Study

The main limitation of our literature study is the limited amount of time available to work on the study and that the literature study study had to be conducted individually. There was a limited amount of time available to work on this study due to it being part of a seminar next to which two other courses were being taken. A such, we could not dedicate all our time to conducting the literature study. Furthermore, although both supervisors proved vital in helping find literature found during snowballing, providing feedback, and pointing out interesting research questions, the literature study had to be conducted and written individually.

6.2 Construct Validity

We may be missing key studies in our literature study due to the inclusion and exclusion being to strict. We could also have missed key studies due to the selection of keywords introducing bias towards certain studies and results. Furthermore, we have the inherent risk of missing certain studies when using the snowballing methodology, as there may be a cluster of literature that we may not of found due to our starting set not covering enough of the literature present.

6.3 Internal Threats

A threat to the internal validity of our literature study, is that the study has been conducted by a single person, rather that multiple people. This is an issue in the search for literature, and selection of literature to include as well as in the data collection on the literature included in the study.

Another threat to the internal validity of our literature study, is how we synthesised the conclusions from multiple studies. These studies have research questions or goals that vary from each other to different degrees. We have tried to overcome this by focusing less on specific cases of the effects of ageing for software developers, as most studies do, and present a higher level overview of the effects of ageing for software developers.

6.4 External Threats

The main external threat we have in this study is that the studies we have found and reported on, although having a broad population when taken together, all have a limited population that they studied. As such, it may not be the case that the common findings we found are valid for software developers as a whole.

7 Summary

In this paper, we conducted a literature study on the effects of ageing for software developers in the work environment. We used the snowballing methodology as presented by Wohlin [25] and the guidelines for systematic literature studies by Kitchenham [8] to conduct the literature study. In total, we found 18 pieces of literature that we used in our literature study. We presented some background on ageing, as well as summarised and synthesised the results of the literature used. We next find that the main limitation of the current literature is the lack of a common definition of what is considered 'older'. We also present possible areas for future research, such as performing longitudinal studies, investigating which measures are most effective to prevent or decrease age stereotypes, and investigating age stereotypes in different cultures. Finally, we presented limitations of our study and threats to validity.

References

[1] S. Baltes and S. Diehl. Towards a theory of software development expertise. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2018, pages 187–200, New York, NY, USA, 2018. ACM.

[2] L. Brooke. Prolonging the careers of older information technology workers: continuity, exit or retirement transitions? *Ageing and Society*, 29(2):237256, 2009.

[3] T. D. Comeau and C. L. Kemp. Intersections of age and masculinities in the information technology industry. *Ageing and Society*, 27(2):215232, 2007.

[4] J. L. Davidson, U. A. Mannan, R. Naik, I. Dua, and C. Jensen. Older adults and free/open source software: A diary study of first-time contributors. In *Proceedings of The International Symposium on Open Collaboration*, OpenSym '14, pages 5:1–5:10, New York, NY, USA, 2014. ACM.

[5] J. L. Davidson, R. Naik, U. A. Mannan, A. Azarbakht, and C. Jensen. On older adults in free/open source software: reflections of contributors and community leaders. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 93–100, July 2014.

[6] Statistisches Bundesamt - Bevlkerung und Erwerbsttigkeit Erwerbsbeteiligung der Bevlkerung Ergebnisse des Mikrozensus zum Arbeitsmarkt. `https://www.destatis.de/DE/Publikationen/Thematisch/Arbeitsmarkt/Erwerbstaetige/ErwerbsbeteiligungBevoelkung2010410167004.pdf?__blob=publicationFile`. Accessed: 2019-01-05.

[7] E. Jovic and J. McMullin. *Risky Business: Aging as an Information Technology Worker*. Routledge, 2016.

[8] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

[9] H. Kleefeld. *Demografischer Wandel und Kompetenz zur Innovation in der IT-Branche — Anforderungen an ein strategisches Human Resource Management*, pages 101–127. Gabler, 2008.

[10] N. Kock, M. Moqbel, Y. Jung, and T. Syn. Do older programmers perform as well as young ones? exploring the intermediate effects of stress and programming experience. *Cognition, Technology & Work*, 20:489–504, 2018.

[11] G. Kowalik and R. Nielek. Senior programmers: Characteristics of elderly users from stack overflow. In E. Spiro and Y.-Y. Ahn, editors, *Social Informatics*, pages 87–96, Cham, 2016. Springer International Publishing.

[12] R. T. Krampe and N. Charness. *Aging and Expertise*, page 835856. Cambridge Handbooks in Psychology. Cambridge University Press, 2 edition, 2018.

[13] U. Lünstroth. *Demography, Aging and High Tech - The Case of Software Developers*. Lit Verlag, 2006.

[14] V. W. Marshall. A life course perspective on information technology work. *Journal of Applied Gerontology*, 30(2):185–198, 2011.

[15] P. Morrison and E. Murphy-Hill. Is programming knowledge related to age? an exploration of stack overflow. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 69–72, May 2013.

[16] P. Morrison, R. Pandita, E. Murphy-Hill, and A. McLaughlin. Veteran developers' contributions and motivations: An open source perspective. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 171–179, Sep. 2016.

[17] CBS - werkzame beroepsbevolking; beroep. `http://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=82808ned&D1=0&D2=0&D3=7-14&D4=120-121,123-124&D5=4,9,14,19,24,29,34,39,44,49,54,59,64,69,74&HDR=G4,T,G1&STB=G2,G3&VW=T`. Accessed: 2019-01-05.

[18] R. A. Posthuma and M. A. Campion. Age stereotypes in the workplace: Common stereotypes, moderators, and future research directions. *Journal of Management*, 35(1):158–188, 2009.

[19] J. Quan, R. Dattero, and S. D. Galup. Impact of age on information technology salaries. In *Global, Social, and Organizational Implications of Emerging Information Resources Management: Concepts and Applications*, pages 403–420. IGI Global, 2010.

[20] U. Schloegel, S. Stegmann, A. Maedche, and R. van Dick. Reducing age stereotypes in software development: The effects of awareness- and cooperation-based diversity interventions. *Journal of Systems and Software*, 121:1 – 15, 2016.

[21] U. Schloegel, S. Stegmann, A. Maedche, and R. van Dick. Age stereotypes in agile software development an empirical study of performance expectations. *Information Technology & People*, 31(1):41–62, 2018.

[22] U. Schloegel, S. Stegmann, R. van Dick, and A. Maedche. Age stereotypes in distributed software development: The impact of culture on age-related performance expectations. *Information and Software Technology*, 97:146 – 162, 2018.

[23] Bureau of Labor Statistics - labor force statistics from the current population survey - 2011. `https://www.bls.gov/cps/aa2011/cpsaat11b.htm`. Accessed: 2019-01-05.

[24] Bureau of Labor Statistics - labor force statistics from the current population survey - 2017. `https://www.bls.gov/cps/cpsaat11b.htm`. Accessed: 2019-01-05.

[25] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 38:1–38:10, New York, NY, USA, 2014. ACM.

[26] United Nations - Ageing. `http://www.un.org/en/sections/issues-depth/ageing/`. Accessed: 2018-12-28.

[27] A. Xia and B. H. Kleiner. Discrimination in the computer industry. *Equal Opportunities International*, 20(5/6/7):117–120, 2001.

Appendices

A Data Collection Starting Set

Age Stereotypes in Agile Software Development- An Empirical Study of Performance Expectations

Authors: U.Schloegel, S.Stegmann, A.Maedche & R.van Dick

Source and full reference: `https://doi.org/10.1108/` [21]
Language: English
Publication year: 2018
Publication venue: Information Technology & People
Population of Interest: Software developers in Agile Software Development (ASD)
Kind of Study: A quantitative survey at the individual level was conducted with 464 employees in two software development (SD) companies.
Variables:

- What age stereotypes are there in ASD?
- Hypothesis 1: Employees in agile software development are biased regarding the performance of younger, middle-aged, and older employees.
- Hypothesis 1a: Performance expectations of middle-aged employees are higher than those of older employees.
- Hypothesis 1b: Performance expectations of middle-aged employees are higher than those of younger employees.
- Hypothesis 1c: The relation between target age and performance expectations is curvilinear in the form of an inverted U shape.
- Hypothesis 2a: Performance stereotypes of older employees in the job role of developer are stronger than performance stereotypes of older employees in ASD in job roles other than developer.
- Hypothesis 2b: Performance stereotypes of younger employees in the job role of developer are stronger than performance stereotypes of younger employees in ASD in job roles other than developer.
- Hypothesis 3: There is an interaction of participant age and target age on performance expectations.
- Hypothesis 3a: Younger participants hold stronger performance stereotypes of older employees than of younger employees.
- Hypothesis 3b: Older participants hold stronger performance stereotypes of younger employees than of older employees.
- Hypothesis 4: There is a three-way interaction of 1) participant age, 2) target age, and 3) job role on performance expectations.
- Hypothesis 4a: Younger employees hold particular strong performance stereotypes of older employees in a job role other than their own.
- Hypothesis 4b: Older employees hold particularly strong performance stereotypes of younger employees in a job role other than their own.

Conclusions:

- Thus, H1 was supported by the data.
- H1a was supported by the data.

- H1b was only partly supported by the data.
- Thus, H1c was only supported for general performance expectations(PE-g)
- Thus, the data supported H2a.
- Thus, H2b was rejected by the data. The bias in PE-g toward younger employees is stronger than the bias in developer performance expectations (PE-D) toward younger developers.
- Thus, H3 was supported by the data.
- Regarding general performance, younger employees favoured middle-aged employees over older employees to a greater extent than older employees did. Regarding developer performance, younger employees favoured middle-aged developers over older developers to a greater extent than older employees did, as hypothesised in H3a. In particular, younger employees held stronger (positive) stereotypes of younger employees concerning PE-D than older employees.
- Thus, H3b was only supported concerning PE-g.
- H4 was partially supported.
- H4a was supported for PE-D.
- H4b was not supported by the data.

Three important results to be highlighted are that: (1) older developers suffer the strongest negative stereotypes in ASD; (2) younger developers hold the strongest bias; (3) the PEs of developers decrease continuously with target age, while the general performance of middle-aged employees (product owner, scrum masters, and development architects) is favoured over that of younger and older employees.

Limitations:

- We used self-report data that might increase common method variance.
- As this was a cross-sectional study, causal relationships cannot be established.
- In addition, the effects of age and generations cannot be separated.

Future research:

- First, the study should be repeated in a few years time because demographics in the software industry change substantially over time.
- Second, additional antecedents for age stereotypes could be examined at an organizational level.
- Third, researchers should add implicit factors (such as attitudes and social cognitions) to ASD-specific theories explaining explicit social aspects.

Age stereotypes in distributed software development: The impact of culture on age-related performance expectations

Authors: U. Schloegel, S. Stegmann, R. van Dick, & A. Maedche

Source and full reference: `https://doi.org/10.1016/j.infsof.2018.01.009` [22]
Language: English
Publication year: 2018
Publication venue: Information and Software Technology
Population of Interest: Software developers
Kind of Study: A quantitative study with 457 employees in two software development companies in China, Germany, Poland and Bulgaria.
Variables:

- How do national and organizational culture influence age stereotypes concerning performance expectations in DSD?
- H1. We predict a curvilinear, quadratic relationship between target age and performance expectations across national cultures in distributed software development (DSD). Across cultures, employees have higher performance expectations toward middle-aged employees than toward both, younger and older employees.
- H2. National culture moderates the effect of target age on performance expectations. Stereotypes toward older employees are more negative in China and Eastern Europe than in Germany. Stereotypes toward younger employees are more negative in Germany than in China and Eastern Europe.
- H2a. Average team age mediates the relationship between national culture and bias in performance expectations, wherein the relationship between national culture and bias as well as the relationship between average team age and bias are moderated by target age. Average team age is lower in China and Eastern Europe than in Germany and thus, the bias toward older employees is stronger than in Germany, while the bias toward younger employees is lower than in Germany.
- H2b. Contact frequency mediates the influence of national culture on bias in performance expectations, wherein the relationship between national culture and contact frequency as well as the relationship between national culture and bias are moderated by target age. In China and Eastern Europe employees have less contact to older employees than in Germany and thus, the bias toward older employees will be stronger. In Germany, employees have less contact to younger employees than in China and Eastern Europe and thus, the bias toward younger employees will be stronger.
- H3. The organizational culture moderates the influence of target age on performance expectations. Stereotypes toward younger and older employees are more negative in organizations which emphasize control and monetary goals (compared to organizations which value teams).
- H3a. Contact frequency mediates the influence of organizational culture on bias in performance expectations. In organizational cultures that value teams, employees have more contact to younger and older employees and thus, the bias toward younger and older employees is reduced.
- H4. We predict a 3-way interaction of target age, organizational culture and national culture on performance expectations: The negative influence of national culture in China and Eastern Europe on stereotypes toward older employees will be buffered by an

organizational culture which values teams (compared to an organizational culture which emphasizes control and monetary goals). The negative influence of national culture in Germany on stereotypes toward younger employees will be buffered by an organizational culture which values teams (compared to an organizational culture which emphasizes control and monetary goals).

Conclusions:

- H1 was supported concerning PE-G. Thus, H1 was supported concerning the main effect of target age on performance expectations, but for PE-D the direction was different than predicted when comparing middle-aged with younger employees.
- Hypothesis 2 was supported by the data.
- Thus, Hypothesis 3 was supported by the data, except for developer expectations toward younger employees.
- H4 was supported for PE-G and partly for PE-D.
- H2a, the hypothesis was supported concerning bias toward older targets.
- Thus, H2b was supported.
- H3a, the data supported the hypothesis.

Limitations:

- Quantitative research applied across different cultural contexts has its limitations.
- As with any cross-sectional study, the directions of influences remain somewhat speculative and thus, moderators and mediators have to be interpreted with care.
- Except for organizational culture we used self-reported data that might increase common method variance, but at least it cannot artificially increase the interaction effects.
- It cannot fully be ruled out that concerns of social desirability might have influenced participants answers.

Future research:

- It would be good if future research would validate our findings using different methodologies.
- Longitudinal studies should be conducted to underpin our results in the future.
- The study should be replicated within more than two companies, and additional aspects of organizational culture which might influence stereotypes, should be added.

younger ($\leq$ 35 years), middle-aged, and older (>50 years) employees

Do older programmers perform as well as young ones? Exploring the intermediate effects of stress and programming experience

Authors: N. Kock, M. Moqbel, Y. Jung, & T. Syn

Source and full reference: `https://doi.org/10.1007/s10111-018-0479-x` [10]
Language: English
Publication year: 2018
Publication venue: Cognition, Technology & Work
Population of Interest: Computer programmers
Kind of Study: A computer programming experiment involving 140 student participants majoring in technology-related areas with ages ranging from 19 to 54 years.
Variables:

- Do older programmers perform as well as young ones?

Conclusions:

- The results of our analyses suggest that age was positively associated with programming experience and perceived stress, that programming experience was positively associated with programming performance, and that perceived stress was negatively associated with programming performance. A moderating effect analysis suggests that as programming experience increased, the association between perceived stress and programming performance weakened; going from strongly negative toward neutral. This happened even as age was controlled for. When taken together, these results suggest that the widespread perception that older adults are under performers is unwarranted. With enough programming experience, older programmers generally perform no better or worse than young ones.

Limitations:

- Our study involved student participants and thus provides an initial test of our theoretical model. Note that even though the age of the student participants ranged from 19 to 54 years, programming experience ranged only from 0 to 5 years.

Future research:

- We recommend that our study be replicated with non-student participants, preferably professional programmers

Impact of Age on Information Technology Salaries

Authors: J. Quan, R. Dattero, & S. D. Galup

Source and full reference: `https://doi.org/10.4018/978-1-60566-962-5.ch021` [19]
Language: English
Publication year: 2010
Publication venue: IGI Global
Population of Interest: Information technology professionals
Kind of Study: Online survey for data collection followed by quantitative research
Variables:

- Is there wage discrimination towards older employees in the IT industry?
- What are managerial implications of this possible discrimination?

Conclusions:

- Age group 30...39 has the highest positive salary bias (0.122), followed by the Age Group 40...49 (0.112), the Age Group 50...59 (0.032), and the Age Group 25...29 (0.016). This clearly suggests that there is wage discrimination towards older employees in the IT industry when compared to the most favourite age group 30...39.
- That is, the employees in the age group 40 and over make only 88% of their younger counterparts when controlling for education and technical experience levels. Therefore, the data suggest that severe age treatment discrimination exists in the IT workforce.
- There is no evidence to support age discrimination in the job functions Networking, QA Testers, and Strategist/Architect.
- for Analyst, DBA, Developer, Project Manager, Support Personnel, and Top Management, and significant for MIS Manager and Web Personnel.
- In terms of wages, the ratios of the typical fit between older and younger ITP are all below 100%, except Top Management with 102%. Therefore, these seem to suggest that age treatment discrimination does not exist homogeneously in all job functions of the IT workforce.
- The results of the study suggest the following: (1) age discrimination exists in overall IT wages, (2) age treatment discrimination does not exist homogeneously in all sectors of the IT workforce, and (3) age treatment discrimination does not exist homogeneously in all job functions in the IT workforce.
- It is essential that management treat all employees fairly during a workforce reduction. They should develop formal policies to address age discrimination. This recommendation is especially meaningful for dynamic industries such as IT, finance, and health care.
- Second, management should create an organisational culture that values older workers and recognises the benefits of their employment. It is important to reevaluate the costs to retrain and renew ITPs skills, due to the rapidly changing IT field, in order to retain the loyalty of older ITP.
- In order to retain the loyalty of older ITP, management needs to pay attention to the changing work attitudes of older ITP and provide opportunities for balancing work-related and non-work-related responsibilities.

Limitations:

- The survey sample was not random since the respondents were self-selecting and voluntary; hence, no-representativeness and the self-report bias are a possibility.

- The survey was on-line which may introduce a bias towards younger employees.
- the survey was placed on an on-line placement companys web site, which may indicate that the survey respondents are more actively seeking new employment than typical ITP and likely tend to sell themselves.

Future research:

- It would be interesting to link age with skills, organization sizes, and gender.
- Investigate the extent of IT age discrimination in different parts of the world.

Intersections of age and masculinities in the information technology industry

Authors: T. D. Comeau, & C. L. Kemp

Source and full reference: https://doi.org/10.1017/S0144686X06005605 [3]
Language: English
Publication year: 2007
Publication venue: Cambridge University Press
Population of Interest: Male IT employees
Kind of Study: Analysis of 76 semi-structured qualitative interviews with male IT workers in two urban areas in Ontario, Canada, in small firms that employed between five and 21 workers.
Variables:

- What are intersection of age with technical masculinities in IT?
- In what ways does IT workers discourse inform and reflect occupational trajectories and tasks?

Conclusions:

- The ability to learn and be up to date was linked to the nimbleness of youth. Age or growing older was understood as the antithesis, that is becoming slow, awkward and ill-suited to the field of play.
- The desire to stay a developer itself suggests a lack of drive, and being driven means moving beyond the programming role.
- Stepping away from technology is a fundamental part of the normal IT career trajectory.
- The natural path as one grows older in IT is to move out of programming and into a managerial role. Ageing ones way out of programming is viewed as maturing or growing up.
- Indeed, workers aged in the mid-thirties were described as seasoned and veterans, depending on how long they had worked in the field, not their chronological age. If a young man started work at 20 years-of-age, then he might be considered a veteran at age 27 years.
- In the discourse about pushing older workers away from programming, the physicality of mental labour was emphasised. The taxing nature of mental labour was seen by many to have a physical component that was portrayed as incompatible with increasing age.
- The notion that older workers are less able to learn (keeping in mind that older means aged in the thirties) may legitimise the removal of older workers from programming work. The evidence from occupational and cognitive psychology does not support a simple cognitive decline explanation.
- Mens priorities may change over time in such a way that the needs of the team no longer takes precedence over any other facet of life.
- We argue that the age-differentiation of roles in the IT industry may disadvantage older workers, and perpetuate ageist ideologies at the organisational level. For those workers who would like to remain in technological work, either full-time or in a reduced or adjusted capacity, the industry provides few options.

Limitations:

- None mentioned

Future research:

- None mentioned

Older Adults and Free/Open Source Software: A Diary Study of First-Time Contributors

Authors: J. L. Davidson, R. Naik, U. A. Mannan, A. Azarbakht, & C. Jensen

Source and full reference: `https://dl.acm.org/citation.cfm?id=2641589` [4]
Language: English
Publication year: 2014
Publication venue: OpenSym '14 Proceedings of The International Symposium on Open Collaboration
Population of Interest: Technically experienced older adults (Aged 50 and older)
Kind of Study: Literature review of the research on the FOSS joining process, motivations, barriers to joining FOSS projects, and research on daily diary studies.
Variables:

- RQ1. Does participation in a FOSS project impact participants self-efficacy?
- RQ2. What motivates older first-time contributors?
- RQ3. What are the benefits and barriers faced by older first-time contributors?
- RQ4. What is the natural contribution process for older first-time contributors?

Conclusions:

- RQ1: Participant 1 who had the highest amount of objective success (discussed in Section 4.4) also had the highest drop in self-efficacy. Participant 3 dropped out of the study and became unreachable. The other two participants showed relatively little change in their self-efficacy of contributing to FOSS. Overall, there is no consistent trend in their self-efficacy scores.
- RQ2: The participants from the daily diary study only reported three motivations: Intrinsic Motivation, Altruism, and Internal Values. Surprisingly, we only found 3 motivations in the current study, however the motivations align well Davidson et al.s findings. One reason for not finding community identification as a motivation in the current study may be because first-time contributors have not realized the full potential of FOSS for making friends and developing a personal community.
- RQ3: Three benefits of contributing to FOSS were Personal Project Need, Free/Cheap, and Mental Challenge.
- RQ3: Participants noted 8 different personal barriers, with the top 2 barriers as Competing Obligations and No Time. Competing obligations included reasons for not participating that were related to being busy in specific ways, such as part-time work, meetings, etc.
- RQ3: We uncovered 9 project barriers. The top 2 barriers were lack of communication and installation issues. A close third was documentation issues.
- RQ4: Another finding from this work was the creation of a preliminary contribution process diagram. See figure 2 in paper.
- RQ4: A major takeaway from the observed contribution process is that communication is vital for our participants to make a successful contribution. First, its important for the participant to ask for help in the correct way. Second, its important for the project to respond in a helpful way. Additionally, like with the photography related application, its important to value contributions other than code, to allow people from all backgrounds to contribute. However, as with Participant 39, there should be pathways for people to contribute in any way they want, including code.

Limitations:

- Small sample size.

Future research:

- Suggest future researchers conduct a large-scale version of this study and include multiple demographics (not just older adults) to continue learning about the FOSS contribution process.
- Model created can be validated by further research into this area.

On Older Adults in Free/Open Source Software: Reflections of Contributors and Community Leaders

Authors: J. L. Davidson, R. Naik, U. A. Mannan, A. Azarbakht, & C. Jensen

Source and full reference: `https://ieeexplore.ieee.org/document/6883029` [5]
Language: English
Publication year: 2014
Publication venue: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)
Population of Interest: Older adults in FOSS (Older = at least 50 years old)
Kind of Study: Literature review, Interview (with 11 older FOSS contributors, aged 50 or over, and 6 FOSS community leaders (of any age))
Variables:

- RQ1: What roles and motivations do older adults have in FOSS communities and how do these change over time?
- RQ2: What are the benefits and challenges facing older adults? This will provide insight on how to highlight benefits and reduce challenges in future outreach efforts.
- RQ3: Do older adults experience or witness discrimination in FOSS communities? There may be something about FOSS communities that deter older adults from contributing.
- RQ4: Do older adults offer anything to FOSS communities that is different from their younger counterparts? If so, what? To encourage FOSS communities to be more welcoming of older participants, it is important that both groups understand the value that older adults bring.

Conclusions:

- RQ1: Thus, our findings align better with the idea of an Onion Patch, where contributors transfer skills from one project to another, and may specialise around certain tasks, than the Onion model.
- RQ1: The top 3 motivations for older participants were intrinsic motivation (10 of 11), community identification (9 of 11), and altruism (9 of 11).
- RQ2: There were multiple benefits mentioned by older participants. Related to community identification, they found that community was a benefit. Satisfaction, widespread use, improved skills, and ease of use were other benefits. Additionally, we see a benefit in participants self perceptions of ageing.
- RQ2: Older participants faced more technical challenges on first contribution, and more social challenges in more recent contributions.
- RQ3: 8 of 11 older adults mentioned witnessing discrimination against non-native English speakers in FOSS communities. 7 of 11 older adults mentioned gender discrimination. No one mentioned experiencing or witnessing ageism.
- RQ4:
  - Software development experience
  - Understanding technology trends
  - Life experience as a user, parent, spouse
  - Experience in general
  - Maturity
  - Understanding computer/software architecture
  - They may have more time

  - They may have more connections/networking
  - Wisdom
  - Professional experience in general

Limitations:

- The sample size was small, which is an artefact of the small population we are studying and the exploratory nature of our research.
- Participants were on the young-old spectrum.
- We did not interview technically experienced older adults who are not contributors for contrast.

Future research:

- None mentioned

Prolonging the careers of older information technology workers: continuity, exit or retirement transitions?

Authors: L. Brooke

Source and full reference: `https://doi.org/10.1017/S0144686X0800768X` [2]
Language: English
Publication year: 2009
Publication venue: Cambridge University Press
Population of Interest: IT workers
Kind of Study: Case study (71 IT workers ages 20-60 with both men and women, and from 10 small and medium-sized IT firms as part of the cross country Workforce Ageing in the New Economy project)
Variables:

- The first theme identified was normative age-based career trajectories
- The second inter-connected theme was the intensity of IT work
- The third was issues associated with the restructuring of IT career trajectories in the firms.

Conclusions:

- Workers aged in the fifties and sixties were experiencing (and resisting) the pressure to retire.
- Age-based perspectives in the information technology industry compress careers and define experience in terms of possessing marketable skills rather than seniority and so are complicit in setting early-retirement norms.
- The career trajectories of older and younger workers were typified by technical specialisation in the early years and broader management roles in later years.
- At the level of the firm, the duration of an individual career depended on a workers capacity to acquire and apply the latest skills rather than accumulated experience. The restricted opportunities and support for training meant that IT workers could outgrow the small firm. Moreover, the short life of many firms truncated many careers, at least within one company.
- Extended careers were perceived to be incompatible with the intensity of IT work.
- Age-segmentation led to the tendency for older workers to be seen as having obsolescent skills and unable to keep up with new developments. The tight gearing between learning and applying new skills intensified the conflicts between later working life and family responsibilities.
- The speed of change in global IT markets and the need for flexible specialisation also impacted on older workers career choices.
- The view of older workers as less trainable in state-of-the-art IT skills than younger workers was reinforced by age segmentation in innovative IT organisations.
- The episodic nature of IT careers made up of a succession of individual IT projects meant that IT workers had commonly been unable to achieve continuous employment and, by extrapolation, to envisage career continuity as they aged.
- Neither sequences of short projects nor longer-term consultancies created progressive, hierarchical and predictable career paths.
- In a stable, medium-sized consultancy firm that employed around 70 people, the career pathways could branch into technical positions rather than ascend vertically.

- A view widely shared among the respondents was that there was a hiatus between specialist IT skills and management and that this led to disconnected and discontinuous career trajectories.

Recommendations:

- Age-bound views of older workers capabilities in the IT industry should be contested, not least to make better use of their skills and capacities as skilled-labour shortages mount.
- Foresight planning that links industry, labour-market opportunities and skills development is needed to connect discrete work episodes into careers.
- Human resources management policies can bridge the career disruption between IT specialisation, project management and management. Articulating work and life trajectories that bridge the present hiatuses between career stages would also be valuable. Promoting continuity in work could be achieved by wider use of flexible hours, contracting back to firms, and home-based E-work.

Limitations:

- None mentioned

Future research:

- None mentioned

Reducing age stereotypes in software development: The effects of awareness- and cooperation-based diversity interventions

Authors: U. Schloegel, S. Stegmann, A. Maedche, & R. van Dick

Source and full reference: `https://doi.org/10.1016/j.jss.2016.07.041` [20]
Language: English
Publication year: 2016
Publication venue: Journal of Systems and Software
Population of Interest: Software developers
Kind of Study: We will test Hypotheses 1 and 2 in a quasi-experiment in the field, by evaluating an awareness-based workshop. We will test Hypothesis 3 in a field experiment, by evaluating a cooperation- based workshop.
Variables:

- Hypothesis 1: An awareness-based intervention on-the-job, with presentations of innovative software development projects that are held by older employees, can significantly reduce bias in performance and innovation expectations comparing middle-aged to older employees.
- Hypothesis 2: We expect an interaction of job role, time and intervention on bias in performance and innovation expectations comparing middle-aged to older employees. An awareness-based intervention on-the-job, with presentations of innovative software development projects that are held by older employees, can significantly reduce the bias, especially for developers .
- Hypothesis 3: A cooperation-based intervention that ensures that employees of different age groups collaborate can significantly reduce bias in performance and innovation expectations comparing middle-aged to older and younger employees. The bias will be reduced (a) short-term and (b) long-term.

Conclusions:

- There was a preference for middle-aged employees performance and innovation expectations over those of older employees.
- There was no significant interaction effect for bias toward younger employees.
- The data supported the hypothesis for bias in innovation expectations. For bias in developer performance expectations it was marginally significant and for BPE-G-mo it was not significant. Thus, our data partly confirmed Hypothesis 2.
- Younger employees developer performance was favoured over that of middle-aged employees.
- Hypothesis 3 concerning short-term effects was supported by the data for all dependent variables except for BPE-G-my.
- Hypothesis 3 concerning the long-term effect was partly supported by the data.
- Our results show that (1) awareness-based and cooperation-based interventions, grounded in the well-proven contact hypothesis, can significantly reduce negative age stereotypes on developer performance, general performance and innovation expectations, (2) bias toward younger as well as toward older employees can be reduced, (3) bias can be reduced causally and long-term, (4) bias that developers hold can be reduced particularly well, and (5) a presentation-based intervention can be brought to large groups of employees without making it mandatory.

Limitations:

- Could only assess reduction of bias in innovation expectations between two points in time.
- We cannot fully rule out that participants might have given socially desired answers.
- Job role moderated the strength of the intervention effect in the awareness-based workshop, however, in the cooperation- based workshop we did not have enough participating developers to test such an effect.

Future research:

- First, further research could try to reduce shortcomings in the present studies. A long-term effect should be tested regarding the awareness-based workshop. For the cooperation-based workshop we assessed reduction of bias in performance expectations across three points in time.
- Second, the design could be adjusted in order to extend the theoretical and practical contribution.
- Third, the interventions should be replicated.

Note: older employees ($>$ 50 years)

Veteran Developers Contributions and Motivations: An Open Source Perspective

Authors: P. Morrison, R. Pandita, E. Murphy-Hill, & A. McLaughlin

Source and full reference: `https://ieeexplore.ieee.org/document/7739681` [16]
Language: English
Publication year: 2016
Publication venue: IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)
Population of Interest: Veteran developers (at least 15 years of experience in software development and who are at least 40 years of age) in open source.
Kind of Study: A quantitative study of StackOverflow, based on a qualitative study of a panel of veteran developers
Variables:

- How do veteran developers' contributions to software development differ from their younger peers?
- Why do few veteran developers participate in open source?

Conclusions:

- StackOverflow data suggests that knowledge about a technology sometimes improves developers knowledge about a successor to that technology, controlling for age and StackOverflow experience.
- The StackOverflow data suggest that veterans have broader knowledge than their younger peers, but their answers for generalist questions are not significantly better.
- While not conclusive, we found some differences in levels of emotional intelligence between veteran and younger developers.
- Veterans appear less interested in marketable skills than their younger peers.
- Veteran developers appear less likely to participate in online social activities than their younger colleagues.
- Younger developers and veterans did not show a statistically significant difference in altruism.

Limitations:

- Panelists are unlikely to be representative of all veteran developers.
- Other findings may have emerged had the panel continued for a longer time.
- the format allows the more vocal participants to sometimes dominate the discussion.
- Another limitation is that the panel did not include veterans who are regular open source contributors, who likely have additional perspectives.
- The participants on StackOverflow may not be representative of the developer population as a whole.
- Construct validity is an inherent threat. For instance, StackOverflows reputation score likely does not capture all dimensions of developer expertise and is also subject to the whims of the community.

Future research:

- Understand how to keep the highly capable veteran developers in the community.
- While veteran software developers appear to have significant contributions to make to open source, such as through their breadth of knowledge, it is unclear how and where they can best contribute.

B Data Collection Iteration 1

A Life Course Perspective on Information Technology Work

Authors: V. W. Marshall

Source and full reference: `https://doi.org/10.1177/0733464810367791` [14]
Language: English
Publication year: 2011
Publication venue: Journal of Applied Gerontology
Population of Interest: IT workers
Kind of Study: Workforce Aging in the New Economy (WANE). Case studies in small and mid-sized IT sector firms in Australia, Canada, England, the United States, the Netherlands, and Germany.
Variables:

- How people make careers in changing work structures
- Age and generational stratification issues

Conclusions:

- Turning to negative designations, older workers are characterized as set in their ways, less adaptive to change, lacking in innovation, and more expensive to employ than younger workers.
- Older workers are also sometimes seen as reluctant to work long and irregular hours. Workers with family obligations are perceived as older, as are workers who have dependents, mortgages, and so on. A worker is characterized as older if he or she wants a more predictable schedule. These stereotypes make older workers and women appear less suited, on average, for the high-demand work of the technical, 24-7 side of IT, that is, for the writing of code.
- The strongest positive attribution to older workers is, not surprisingly, that they are more experienced.

Limitations:

- None mentioned

Future research:

- None mentioned

Demografischer Wandel und Kompetenz zur Innovation in der IT-Branche Anforderungen an ein strategisches Human Resource Management

Authors: H. Kleefeld

Source and full reference: `https://doi.org/10.1007/978-3-8350-5570-4_7` [9]
Language: German
Publication year: 2008
Publication venue: Gabler
Population of Interest: Older IT workers
Kind of Study: Literature study
Variables:

- Does advancing age have an influence on the individual innovation competence of the employees?
- Will the demographic developments in the societies of the industrialized world inevitably have a negative impact on the innovative capacity of companies?

Conclusions:

- On the one hand, the starting point for the ability to innovate is the knowledge available for problem solving. Older employees will usually have a high level of experience assumed. The role of empirical knowledge in the development of innovations is still being discussed in a contradictory way in labor psychology research.
- In turn, the willingness to learn and ability to learn are of particular interest for the person's ability to innovate. A compilation of diverse studies on learning ability shows that although there are differences in the learning behavior of younger and older persons, these are crucially influenced by a variety of somatic, social, pedagogical and biographical influencing factors.
- Thus, the first challenge for Strategic Human Resource Management is to assess the potential impact of demographic change on the organization's assets.
- With regard to influencing social capital, it is above all the composition of the teams in innovation processes that presents itself as a design task. On the one hand, it is important to make productive use of the skills of all employees in the sense of a "diversity approach". On the other hand, the conscious shaping or control of knowledge transfer plays an even greater role, the more the importance of empirical knowledge decreases and large cohorts will retire for reasons of age.
- Strategic Human Resource Management therefore has the task of correspondingly influencing the age image and the corporate culture through appropriate communication measures so that the potential of older employees can be sustainably used for the competitiveness of the company.
- In addition to identifying and developing individual strategic competencies, Strategic Human Resource Management has the task of ensuring that employees have the opportunity and responsibility to participate in innovative projects throughout their entire working lives. Human Resource policies on recruitment, assessment, incentive design and human resource development should be designed accordingly.

Limitations:

- None mentioned

Future research:

- None mentioned

Demography, Aging and High Tech - The Case of Software Developers

Authors: Uwe Lünstroth

Source and full reference: `http://www.litwebshop.de/index.php?main_page=product_info&products_id=2171` [13]
Language: English
Publication year: 2006
Publication venue: LIT Verlag
Population of Interest: Software developers in Agile Software Development (ASD)
Population of Interest: Older software developers (> 40 years old)
Kind of Study: Case study and interview with 29 experts
Variables:

- Which causes can be found hindering the older engineer to participate in innovative work and under which conditions should it be possible to retrain these people?
- What are possible measures to overcome such difficulties?

Conclusions:

- Situation in case study 1: The organisational structure war only designed to fit to the purpose of current demands and not to the development of peoples qualifications for further demands. Collaboration and learning was restricted to special fields of word without taking into consideration the possibilities of technical and organisational change. There was only little time for possibilities in self-development along with the changing technology and new paradigms and concepts in software programming and hardware design.
- Situation in case study 2: Older developers were allowed to retrain and learn in the new fields of technology. There is no obvious negative image of the older worker regarding their ability to take place in innovation processes. This allows for team collaboration between younger and older software developers, self-development in changing tasks and continuous learning, both within the team and on abilities.
- Companies in the INVAS and SUFAW case studies show a general point of deficiency in personnel management.
  - There is no long term planning of fields of activity and the according activities for enhancement of qualification.
  - The personnel management is not acquainted with older developers and their possible problems of career development.
- Especially in the phases where social competence or experience in projects is of importance, the work should be open to older developers.
- The older developer, who has more experience and a greater social ability, should be involved in first phases and end phases of the development process
- Career planning should allow for a wide range of possibilities in performing tasks during different life stages.
- Career planning approach and aims is presented in Chapter 5.2.
- The personnel management should give priority to sustaining and maintaining the knowledge of the developer in connection with the special tasks of the company.
- Older developers should be brought towards certain fields.

Older software developers do not need recreation positions. They have to widen their abilities and task range in the company in steps that can be planned in some general aspects for periods

of a decade and further on.

Limitations:

- None reported

Future research:

- None reported

Discrimination in the computer industry

Authors: A. Xia, & B. H. Kleiner

Source and full reference: https://doi.org/10.1108/02610150110786859 [27]
Language: English
Publication year: 2001
Publication venue: Equal Opportunities International
Population of Interest: Workers in the Computer Industry
Kind of Study: Literature survey
Variables:

- Why Discriminate - An Analysis of the Causes and Consequences
- Economic Incentives Boost Discrimination
- Consequences of Discrimination - Intel under the Fire

Conclusions:

- A youth-oriented Industry. The computer industry is a new and fast-growing industry. People who work in this industry have to update their skills and knowledge constantly in order to keep abreast with the rapidly new technology development.
- Run by the Young leader. Software companies and Internet start-ups, in particular, tend to be founded and run by young people who are simply more comfortable working with their peers.
- Experience is not the Most Valued Asset.
- Cost effective in hiring young employees.
- Consequences: Labour shortage, Legal actions and Intense relationship.

Limitations:

- None mentioned

Future research:

- None mentioned

Is Programming Knowledge Related To Age? An Exploration of Stack Overflow

Authors: P. Morrison, & E. Murphy-Hill

Source and full reference: `https://ieeexplore.ieee.org/document/6624008` [15]
Language: English
Publication year: 2013
Publication venue: 10th Working Conference on Mining Software Repositories (MSR)
Population of Interest: Users of Stack Overflow between 15-70, active in 2012, and reputation between 2-100000
Kind of Study: Case study, quantitative research of SO data
Variables:

- RQ1: Does age have a positive effect on programming knowledge?
- RQ2: Do older programmers possess a wider variety of technologies and skills?
- RQ3: To what degree do older programmers learn new technologies?

Conclusions:

- SO data suggests that there is a positive relationship between age and reputation on SO.
- There is initially a decline in the mean number of tags per programmer, bottoming around age 30, followed by an increase in the 40s and 50s and dispersion in the 60s.
- Given the strength of the relationship between age and the selected new technologies is relatively weak, we do not have strong evidence against older programmers learning new technologies. It appears that older programmers do learn new technologies.
- Correlation between age and SO reputation, which may indicate that programming knowledge can be maintained at a high level in to a persons 50s and 60s. It appears that older SO users not only can acquire additional knowledge, but that they acquire knowledge of new technologies, in the case of the technologies we have examined.

Limitations:

- Does the SO population represent the programmer population?
- Does SO reputation measure programming knowledge?
- Does measuring programming knowledge say anything about programming ability?
- We are not convinced that our means for answering RQ3 is fair, although we do not yet have a better procedure.

Future research:

- If the trend shown in the data of programmers leaving the profession before they have fully developed were present in the software development industry, slowing it would produce an effective increase in the number of trained programmers available for developing software. Studying and acting on this information at the organizational level might call for new perspectives on the part of boards, executives, management and human resources in recruiting and promotion. At the individual level, awareness of high performance on the part of others can inspire efforts to continue or to improve.
- Research into the nature of the SO population and its relation to the programmer population at large needs to be conducted, in order to support inferences to the programmer population. Further investigation is needed of how SO measures, such as user reputation and question scores translate in to programming knowledge and ability.

C Data Collection Iteration 2

Senior Programmers: Characteristics of Elderly Users from Stack Overflow

Authors: G. Kowalik & R. Nielek

Source and full reference: `https://doi.org/10.1007/978-3-319-47874-6_7` [11]
Language: English
Publication year: 2016
Publication venue: Springer
Population of Interest: Elderly users of Stack Overflow (60 or more years old)
Kind of Study: Quantitative research. Analyisis of two data sources: Stack Exchange API, Stack Overflow Survey
Variables:

- Do Older Adults Use Stack Overflow?
- Do They Teach or Do They Learn?
- Earned Reputation or Gamed Reputation?

Conclusions:

- After excluding users with too high age, we received 562795 users (0.8% of them have a age 60 or more). This is very similar to the survey results in 2016 (There is 0.5% 0.8% of users with age 60+.).
- This confirms our hypothesis seniors more often (more than 10 percentage points) gives answers, share their knowledge, than ask questions. They more teach than learn.
- This confirms our hypothesis that they put more effort into their activity.
- We also observe the differences in motivation with older adults biased more toward sense of responsibility but because of small sample differences are not statistically significant.
- Older adults have higher reputation (significance test confirms it). Seniors have higher mean of reputation, but also much higher standard deviation.

Limitations:

- None mentioned

Future research:

- None mentioned

Towards a Theory of Software Development Expertise

Authors: S. Baltes & S. Diehl

Source and full reference: `https://doi.org/10.1145/3236024.3236061` [1]
Language: English
Publication year: 2018
Publication venue: ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering
Population of Interest: Software Engineers
Kind of Study: Quantitative research and online questionnaire, which we sent to a random sample of software developers.
Variables:

- Contribute a theory that describes central properties of software development expertise and important factors influencing its formation. Our goal was to develop a process theory, that is a theory intended to explain and understand how an entity changes and develops over time. (Note that we will focus on age factors influencing its formation)

Conclusions:

- We asked participants if they ever observed a significant decline of their own programming performance or the performance of co-workers over time. Combining the answers from S2 and S3, 41.5% of the 205 participants who answered that question actually observed such a performance decline over time. We asked those participants to describe how the decline manifested itself and to suggest possible reasons. The main categories we assigned to those answers were: 1. different reasons for demotivation (34), 2. changes in the work environment (32), 3. age-related decline (13), 4. changes in attitude (10), and 5. shifting towards other tasks (7).
- Age-related decline was described in both samples, but the more elaborate answers came from the experienced developers.

Limitations:

- Analysis of expertise self-assessments, we cannot rule out that a confounding factor lead to the higher self-assessments of experienced developers.
- Focus on Java and on open source software development, in particular on GH and SO users.
- The qualitative analysis and theory-building was mainly conducted by the first author and was then discussed with the second author.

Future research:

- Researchers can use our methodological findings about (self-assessed) expertise and experience when designing studies involving self-assessments.
- One could review studies on developers motivation and personality in the context of our theory

Appendix B

Code Book

Codes:	short description – the name of the code itself	inclusion criteria – conditions of the datum or phenomenon that merit the code	typical exemplars – a few examples of data that best represent the code	atypical exemplars – extreme or special examples of data that still represent the code
	RQ1: General acknowledgement of ageism in software industry	Acknowlegding age as limiting, admitting there is ageism etc. Also in tech industry as a whole	Silicon Valley has a reputation for favoring younger engineers, developers, and programmers.	With age and experience comes wisdom, or so the saying goes, but when it comes to being a developer is it a job you can pursue into middle age?
	RQ1: Use of authority figures	Mentioning, or implying thoughts of, a famous or influential person in tech, or person in a postion of authority	Facebook CEO Mark Zuckerberg famously said back in 2007 at Stanford: "Young people are just smarter."	Prominent figures in tech have reiterated this point in the past.
	RQ1: Use of related studies	Using a study on the topic, note that these can be both scientific and non scientific studies	On the heels of research that found your chances of getting hired in Silicon Valley plummet after the age of 48, a new study has found that nearly half of those already working in the industry fear getting the ax because of their advancing age.	This is not openly discussed, because employers could be accused of age discrimination. But research, such as that completed by University of California, Berkeley, professors Clair Brown and Greg Linden shows that even those with masters degrees and Ph.Ds have reason to worry.
	RQ1: Use of related articles	Using an article on the topic, either a reference or a summary of the article	But the problem extends beyond the hiring preferences of today's tech companies. Some developers feel stifled by the time they hit age 30, as game designer Michael O. Church [wrote in a blog post] from 2012.	Lots of links to Quora questions along the lines of "Is _ too old to start as a developer"
	RQ1: Demographics		According to the data, the average developer is 28 years old, as the chart from Stack Overflow shows: []	Statistics show that most software developers are out of the field by age 40, Matloff continues, and here my eyebrows really start to rise. Most programmers? As in the majority of them? Gone? (Matloff declines to mention which statistics he's reading.)
	RQ1: Use of related forum posts	Using a forum post on the topic, or a summary of forum post(s)	An anonymous user recently asked on Quora: “I’m 35 years old. Am I too old to join Google, Facebook, Microsoft or Apple as a software engineer?” Quora is a website that allows people to post questions and have users answer them.	Lots of links to Quora questions along the lines of "Is _ too old to start as a developer"
	RQ1: Reproduction of discorse - Other	Any reproduction of discourse that does not fit any existing code		

Codes:	short description – the name of the code itself	inclusion criteria – conditions of the datum or phenomenon that merit the code	typical exemplars – a few examples of data that best represent the code	atypical exemplars – extreme or special examples of data that still represent the code
	RQ3: Personal experiences	Personal experience by the author (or whomever is speaking), or reference to personal experience of someone else	I spent my 20s working as a teacher. I didn't learn to code until I was 30.	I knew several people who were much older than me when they got their first developer job.
	RQ3: Speculation	Speculation by the author or others on age related aspects in software development	Of course, the data isn't an absolute indication of the tech workforce in Silicon Valley or any other area of the world — it just provides a bit more evidence to the argument that younger developers are in higher demand.	But who cares what Matloff says anyway, right? Lies, damn lies, statistics, and all that. Older tech workers probably have a fair idea of where they are in their careers, despite the doomsayers. What troubles me, though, is the message that articles like Matloff's send to the younger generation, particularly those who have yet to enter the workforce.
	Other (Please add code in comments)	Any paragraph that contains usefull information but for which there is no existing code		
	N/A	Not applicable for our study		
	RQ2: Social challenges	Age related aspects that present social challenges for software developers, older or younger. These can be related to social situations at work or outside of work	"Although ageism is rare, you may feel out of place at times since most of your coworkers will be much younger than you are," he wrote. "This comes up more often in social situations than in technical ones."	If you're a technology professional looking for a company that will help you feel superhuman, ask what it's building. Join a [business that takes risks] and is willing to invest. I sat with a business that was trying to build a product, and it was told that it couldn't happen. But it did it, and it's growing fast. That's an exciting journey and bound to [make software developers feel young], regardless of their birth certificates.
	RQ2: Professional challenges	Age related aspects that cause professional challenges for software developers, such as finding a job or being able to keep a job. Challenges that are caused by age/the perspective of age	Facebook CEO Mark Zuckerberg famously said back in 2007 at Stanford: "Young people are just smarter."	"I'm ____ years old. Am I too old to get hired as a developer?"

Codes:	short description – the name of the code itself	inclusion criteria – conditions of the datum or phenomenon that merit the code	typical exemplars – a few examples of data that best represent the code	atypical exemplars – extreme or special examples of data that still represent the code
	RQ2: Career path	Things that occur, or are thought to occur, during a software developers career. Such as promotion, increasing salary, repeating similar work	Other programmers are inevitably [promoted to management]. I know, you're rolling your eyes. "A career in software development doesn't necessarily prepare you to be a great manager," you're saying. Guess what? Neither do management training courses.	If all you do is 'write code' then you have to be prepared to 'write the same code' in a new paradigm several times, one commenter, ChuckMcM, wrote.
	RQ2: Reported aspects - Other	Any age related aspect reported that does not fit any existing code		