UNDERSTANDING AND SUPPORTING CURATION IN THE SOFTWARE DEVELOPERS’ COMMUNITY

A Dissertation in
Information Sciences and Technology

by
Yu Wu

© 2018 Yu Wu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2018
The dissertation of Yu Wu was reviewed and approved* by the following:

John M. Carroll  
Distinguished Professor of Information Sciences and Technology  
Dissertation Advisor  
Chair of Committee

Xiaolong (Luke) Zhang  
Associate Professor of Information Sciences and Technology

Steven Haynes  
Associate Teaching Professor of Information Sciences and Technology

S. Shyam Sundar  
Distinguished Professor / Co-Director, Media Effects Research Laboratory

Audris Mockus  
Ericsson-Harlan D. Mills Chair Professor of Digital Archeology and Evidence Engineering in the Department of Electrical Engineering and Computer Science of the University of Tennessee

Jessica Kropczynski  
Assistant Professor, School of Information Technology, University of Cincinnati

Andrea Tapia  
Associate Professor of Information Sciences and Technology  
Director of Graduate Programs

*Signatures are on file in the Graduate School
ABSTRACT

GitHub has become a crucial part of the software developers’ community, where millions of software developers from all over the world come together to share source code and collaborate on software projects. In recent years, appropriating GitHub repositories for curation purposes has become popular. Particularly, software developers have started to adopt GitHub repositories and other features to collect, evaluate, organize, and preserve the Internet resources. Currently, there is an inadequate amount of literature available that examines how well GitHub features are supporting this and how this practice is influencing the software developers’ community.

In this thesis, we study curation repositories as a new category of GitHub repository to understand: (1) how GitHub features support this practice; (2) what motivates software developers to curate resources, and especially why GitHub is chosen; (3) how curated resources are used by software developers, and how the GitHub could better support curation.

We first conduct statistical analysis on GitHub activity data as well as content analysis on the popular curation repositories in 2014. We compare and contrast practices in curation repositories with software repositories. Results show that (1) the curation category has quickly become popular among repositories on GitHub, (2) curation is directed at learning and professional development, and (3) the curation practice leverages collaborative tools and practices native to GitHub in new ways. Although curation and software repositories use the same set of activities for development, they are different from each other in terms of the quantity of each type of activity performed by developers. Our results suggest that curation is becoming increasingly important to GitHub users and that current curation practices can be better supported with tools designed specifically for curation.
Next, we conducted in-depth interviews with 16 software developers, each of whom hosts curation projects on GitHub, to understand curators’ experiences with curation on GitHub. Our results suggest that the motivators that inspire software developers to curate resources on GitHub are similar to those that motivate them to participate in the development of open source projects. Convenient tools (e.g., the Markdown syntax and Git version control system) and the opportunity to address professional needs of a large number of peers attract developers to engage in curation projects on GitHub. Benefits of curating on GitHub include learning opportunities, support for development work, and professional interaction. However, curation is limited by GitHub’s document structure and format and by a lack of search function. In light of this, we propose design possibilities to encourage and improve appropriations of GitHub for curation.

Last, we did a survey study on the users of curation repositories and found out that software developers perform multiple types of information discovery behaviors to visit curation repositories to look for resources that are distributed all over the Internet for learning, supporting work, and following trends. Information cues in GitHub play important roles for communicating the quality of curated resources. The results informed the design and implementation of RepoHunter, which directly attaches information cues to curated items in a curation repository. The evaluation of RepoHunter reveals its improvement of user experiences and suggests further design opportunities.

This thesis sheds lights on the significances of curation in supporting the software developers’ community, the role of curators in helping the community filter and prioritize resources, and the technology, i.e. GitHub, that host and communicate knowledge to a wide population. It also calls for future research direction in supporting curation as a practice and in deepening our understanding of software developers’ information behavior in the social media era.
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... viii

LIST OF TABLES .............................................................................................................. ix

ACKNOWLEDGEMENTS .................................................................................................... x

Chapter 1 Introduction ...................................................................................................... 1

  Appropriating GitHub for Curation ................................................................................ 4
  Dissertation Objectives Scope ......................................................................................... 6
  Dissertation Structure ..................................................................................................... 8

Chapter 2 Foundations ..................................................................................................... 10

  Communities of Practice ............................................................................................... 10
  Technology and Communities of Practice .................................................................... 11
  The Software Developers’ Community .......................................................................... 12
    Early Technologies that Support the Software Developers’ Community ................. 13
    Social Media and Software Developers’ Community ................................................ 13
  Challenges Software Developers Facing in the Era of Social Media ......................... 16
  Curation .......................................................................................................................... 18
    Curation Activity .......................................................................................................... 19
    Curator .......................................................................................................................... 20
    Curation in Social Media ............................................................................................... 21
    Curation Repositories on GitHub ................................................................................ 24
  Gaps .................................................................................................................................. 25

Chapter 3 Related Studies ............................................................................................... 27

  Curation in the GitHub Environment ........................................................................... 27
    GitHub Repository ........................................................................................................ 27
  RQ1: How Does the GitHub Context Support Curation? .............................................. 28
    The Purposes of Curation in GitHub ........................................................................... 28
    The Curation Leadership and Content ....................................................................... 29
    The Collaboration Patterns in Curation Repositories ................................................. 30
  RQ2: What Motivates Software Developers to Curate on GitHub? ............................ 31
    Motivations to curate in the social media era ............................................................ 31
    Software developers’ motivations for participating online communities .................. 33
  RQ3: What Are the Software Developers’ Experiences with Curation from a User’s
    Perspective ................................................................................................................... 34
    Information Discovery Framework .............................................................................. 34
    Motivations in Information Discovery ....................................................................... 36
    Information Cues .......................................................................................................... 36

Chapter 4 How GitHub Supports Curation Repositories ................................................. 39
Method ........................................................................................................... 41
Results ........................................................................................................... 43
  Curation Repository vs. Software Repository ........................................... 43
  The Emerging Role of Curation Repositories ........................................... 46
Discussion ...................................................................................................... 53
  Internal Curation on GitHub ................................................................. 53
  Implications for the Owners of Curation Repositories ......................... 54
  Collaborative Curation on GitHub ......................................................... 55
Design Implications ....................................................................................... 57
Limitations ..................................................................................................... 57
Summary ........................................................................................................ 58

Chapter 5 Curators Motivations and Experiences ...................................... 59
Methodology .................................................................................................. 60
  Participants Recruitment .......................................................................... 60
  Interview Protocol .................................................................................... 61
  Data Analysis Procedure ......................................................................... 61
Results ........................................................................................................... 62
  Motivations for Curation .......................................................................... 63
  The technology choice—why GitHub ...................................................... 66
  The use of curated resources ................................................................... 68
  Limitations of using GitHub for curation ............................................... 71
Discussion ..................................................................................................... 73
  The Extrinsically Motivated Curators ...................................................... 74
  Leveraging GitHub as a tool for communication ..................................... 75
  Curation to strengthen software developers’ community ......................... 76
  Design Implications and future directions ............................................. 77
  Limitations ............................................................................................... 78
Conclusion ..................................................................................................... 80

Chapter 6 Understanding and Supporting User Experiences with Curation Repositories on GitHub ......................................................... 81
A Case Study to Understand the Curation Repository ................................ 82
  Participants and Procedure ...................................................................... 82
  Data Analysis and Results ....................................................................... 83
  Reasons for Visiting Curation Repositories .......................................... 84
  Types of Information Discovery Behaviors in Curation repositories ......... 85
  Information Cues for Evaluating Curated Items ...................................... 87
RepoHunter: Bringing Information Cues to Curated Items to Support Curation .................................................................................. 89
Results ........................................................................................................... 92
Discussion ..................................................................................................... 94
  Information Discovery in Curation Repositories .................................... 94
RepoHunter Implications ............................................................................. 95

Chapter 7 Discussion ..................................................................................... 97
Supporting the Software Developers’ Community with Curation......................... 98
The Curator Role.................................................................................. 101
The Social Coding Service for Curation..................................................... 102
Future Directions.................................................................................... 103
  Supporting Curation Community ........................................................... 103
  Software Developers’ Information Needs in Social Media......................... 104

Chapter 8  Conclusion............................................................................... 106

References ............................................................................................. 108
LIST OF FIGURES

Figure 1-1: Example of curated list of software development resources (free-programming-books on GitHub)........................................................................................................... 5

Figure 1-2. Summary of the research questions and the scope of this dissertation. ............ 8

Figure 2-1. A part of the README.md file of the awesome-go curation project............. 24

Figure 4-1. Average number of stars received by curation repositories and software repositories.................................................................................................................. 43

Figure 4-2. The repository counts vs. the percentage of Type Add, Delete and Update of pull requests. ............................................................................................................. 52

Figure 6-1(a). The original version of the awesome-go repository........................................ 89

Figure 6-1(b). The RepoHunter version of the awesome-go repository, including the number of stars and last pushed date for each curated item................................................. 90
LIST OF TABLES

Table 4-1. Independent sample t-tests that compare the log-transformed mean of the number of seven types of activities between curation projects and software projects. ... 45

Table 4-2. Types of curated resources. ................................................................. 47

Table 4-3. Sources of curation repositories. ....................................................... 48

Table 4-4. Levels of description in curation repositories. ................................. 48

Table 4-5. The number of GitHub users following curation repository owners prior to the creation of curation projects. ................................................................. 49

Table 4-6. Number of contributors to curation repositories. ............................ 50

Table 4-7. Percentage of Pull Requests that have more than 2 participants involved. .... 51

Table 5-1. Coding scheme summary. ................................................................. 62

Table 6-1. Serendipitous discovery vs fact discovery ....................................... 86

Table 6-2. Frequency of rediscovery for awesome-go curation repository .......... 86

Table 6-3. Information scent and associated cues software developers look for when evaluating curated items. ................................................................. 88
ACKNOWLEDGEMENTS

I would like to express my deep appreciation and thanks to my advisor Dr. John M. Carroll, who has been a tremendous mentor for me. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. His advice on both research and my career have been invaluable. He has taught and distilled me, both consciously and unconsciously, to see the bigger picture of problems and to steer myself towards the goals to achieve. I live with the principles I learned from him every day. I appreciate all his contributions in time, ideas, and funding to make my Ph.D. experience productive and stimulating. I am also thankful for the excellent role model he served as a scholar. The joy and passion he has for his research were contagious and motivational for me, even during tough times in the Ph.D. pursuit.

I would also like to thank my committee members, Dr. S. Shyam Sundar, Dr. Xiaolong (Luke) Zhang, and Dr. Steven Haynes, for serving as my committee members as well as for guiding me towards finishing my thesis. I took several Dr. Sundar’s classes where rigid scientific methodology was emphasized. Dr. Zhang not only had given me effective and pragmatic suggestions on my thesis, but also cared for my wellbeing. Dr. Hayes has taught me to look at the current research problems in the software engineering from historical lenses, which was inspiring during the time when I was stuck.

I also thank Dr. Audris Mockus, not only for serving as the special member of my committee, but also for offering me the research internship opportunity at Avaya Labs, where he coached me with big data / quantitative methods in the enterprise environment. The lessons learned during the summer of 2014 in Basking Ridge, NJ, have given me strategic advantages in my career already.
Special thanks go to Dr. Jessica Kropczynski, who was the go-to person if I had any difficulties with my thesis and my research in general. She has been a co-author for a major part of my publications and has contributed a considerable amount of her time and energy to help my research.

I also thank Dr. Mary Beth Rosson, who has been a warm host of both the CSCL/HCI lab and the College of Information Sciences and Technology. It is not easy for foreign students to pursue a graduate degree and to make a living in an unacquainted environment. But Dr. Rosson was always kind to every one of us, listened to the difficulties and challenges we had, and helped us as much as she could. She made us feel a lot closer to our home.

Finally, I would like to thank my family. My parents have always encouraged me to pursue my dream and myself, supporting me unconditionally with love and care. I am forever in debt to my wife, Na Wang, whom I met at a party hosted by Dr. Carroll and Dr. Rosson. She took care of a significant part of the trivial matters of our lives so that I could spend more time on my thesis. As an HCI scholar herself, she also helped me tremendously with my research. Last but not least, I thank Yuna, our daughter, who was born on 2/26/2018, and has been a lovely angel to our lives since then.
Chapter 1

Introduction

In recent years, social media have gained ample application and played an increasingly important role in the software developers’ community. Many software development-oriented social media have facilitated and enhanced the way software developers communicate, collaborate, and learn (Singer et al., 2013). For instance, recent research literature suggests that social media sites, such as Twitter, Stack Overflow, and GitHub, are being increasingly adopted in the software developers’ community, developing new ways for software developers to make connections with others, to ask for help, and to collaborate with others (Singer et al., 2013; Singer et al., 2014; Storey et al., 2014).

Twitter is prevalently used as a tool to receive the latest news and make contact with others (Bougie et al., 2011), and it is used by software developers to post and follow the newest trends in the tech industry, make connections with like-minded, and discuss development-related topics (Singer et al., 2014). Stack Overflow serves as a Q&A platform where developers can not only ask and answer questions, but also search for answers for specific problems encountered during software development. GitHub is a work production platform for software developers to host projects and collaborate with others. It also includes many social features, such as a user profile, a follow function, and activity traces, which are intensively used by software developers to make connections and form impressions about others (Dabbish et al., 2012; Marlow et al., 2013). Software developers have increasingly cooperated on multiple social media sites to engage in software development-related practices. And in the meantime, they have also become dependent on the online resources generated from these sites (Storey et al., 2017).
The booming of social media in software development is not coming without any cost. Too many social media channels that users have to navigate through have already created difficulties for software developers to choose and follow (Storey et al., 2014). On the one hand, the amount of content generated by social media is raising exponentially. For example, on December 2012, there were 4.5 million software repositories hosted on GitHub (Marlow et al., 2012). The number reached 10 million repositories in 2013\(^1\), and then grew to 67 million in 2017\(^2\). At the same time, many of the open source projects on GitHub are constantly updating, evolving, and terminating, making it difficult for software developers to evaluate the quality of a project and to keep track of the active and useful ones. Another example of an explosion of resource generation in the software developers’ community is Stack Overflow. Stack Overflow has generated 2.17 million questions and 3.28 million answers just between 9/2012 and 9/2013\(^3\). If other social media used by developers are taken into consideration, such as Twitter, Hacker News, and Reddit, the amount and rate of new information generated is unimaginable. Individual software developers are unlikely to find relevant resources efficiently from an information repository of this size on their own efforts.

In addition to the volume of content that is spread all over the online space, the quality of the content is of another concern. Researchers have found that the quality of user-generated content varies widely. For examples, Agichtein et al. (2008) found that the quality of content generated on social media sites is of high variance: “... from very high quality to very low quality, sometimes abusive content”. In addition, the majority of user-generated videos is of low interest to most users (Cha et al., 2007).

---

1. [https://github.com/blog/1724-10-million-repositories](https://github.com/blog/1724-10-million-repositories)
2. [https://octoverse.github.com/](https://octoverse.github.com/)
As the participation scope is drastically increasing, the speed of resource contributions across different platforms is soaring, and the concerns about content quality are escalating, it can make finding the best resource to a particular problem an overwhelming and laborious task.

During the same period of time, curation services have also started to gain popularity. Curation is the activity to select, evaluate, organize, and preserve resources for future use (Duh et al., 2012). Recent curation services, such as Pinterest, allow users to create collections of content, such as images, news, and web pages, of their own interest and share these with others (Saaya et al., 2013). For example, Pinterest allows users to “pin” photos they found from different online sources, and organize them into different topics, such as sports, fashion, hobbies, etc. (Gilbert et al., 2013).

Curation is likely to addresses the tensions between the vast quantities of content and the limited attention of users: curators collect content, evaluate the quality of each item, and promote a small set of content that users should focus their attention on (Askalidis and Stoddard, 2013). Human efforts play important roles in the curation practice, where they can capture the nuances in the content and audience groups and add qualitative judgement to content being curated, while information aggregated by computers lacks such a qualitative organization (Rosenbaum, 2011).

Given that the nature of curation is to apply human efforts to select and maintain quality resources of interest, it can be a promising solution to address the emerging challenge in the software developers’ community, which is software developers’ needs to find quality resources in this fast growing information space. Moreover, we recently observed that appropriating GitHub for curation is starting to gain momentum.
Appropriating GitHub for Curation

GitHub is a code repository hosting site, based on the Git version control system. It allows software developers to create software repositories, share with others, and contribute to others’ repositories. It also includes a number of social features that support software practices, such as user profiles, following functions, and activity histories. It is highly welcomed by software developers (Marlow et al., 2013), as a large community of software developers come to GitHub to host and share their own projects, contribute to other open source projects, and follow famous coding “rock stars” (Dabbish et al., 2012).

In addition to hosting and collaborating through the use of GitHub repositories, curation repositories are emerging as a new category of repository, where software developers appropriate GitHub repositories to create public resources lists (Wu et al., 2015). In such a GitHub repository, software developers select, evaluate, and systematically organize resources for preservation and future use. In 2014 and 2015, repositories, such as awesome-python⁴ and awesome-go⁵, which curate resources about topics on certain programming languages, gained much popularity on GitHub. The number of curation repositories has steadily increased since then and many of them have remained among the most famous repositories on the entire platform (Wu et al., 2014).

An example of a curated list created on GitHub is shown in Figure 1-1. “Free Programming Books” is a collection of free learning resources for software developers. It was initially created on Stack Overflow by George Stocker, and later cloned to GitHub by Victor Felder⁶. It contains a comprehensive list of resources for learning different programming languages, which is selected by around 1000 developers according to its project page on GitHub. This repository is well received by the developers’ community; it has been cloned more than

⁴ https://github.com/vinta/awesome-python
⁵ https://github.com/avelino/awesome-go
25,000 times on GitHub. It also received a lot of gratitude on HackerNews\(^7\), such as “It really does save you loads of time as compared to searching for books using Google. – computerjunkie”.

Curation repositories are enabled by GitHub features. They appropriate the README.md file of a repository to create a list of resource indexes within one page. It categorizes resources into different themes and separates them into sections. Typically, each resource is recorded with the resource name and a brief description of the resource (see Figure 1-1). In addition, URLs are attached to each of the curated items. Clicking a resource name (shown in blue in Figure 1-1) will direct the user to the real web location of the resource.

\(^7\) [https://news.ycombinator.com/item?id=7557911](https://news.ycombinator.com/item?id=7557911)

---

**Figure 1-1**: Example of curated list of software development resources (free-programming-books on GitHub).
Curation repositories on GitHub draw a great deal of attention from the software developers’ community. For example, the curation repository of “Free Programming Books” showed above currently has over 100,000 stars, which is the third most starred repository of the entire site, although it was only created in 2013, so it is a lot younger as compared to many popular software repositories on GitHub.

This thesis will aid in understanding and supporting curation in the software developers’ community. Particularly, it will investigate curation repositories themselves on GitHub, software developers who create curation repositories, and software developers who visit curation repositories.

**Dissertation Objectives Scope**

Understanding the reasons for the advent of curation as well as the technology that supports the practice can be helpful in supporting the software developers’ community better as well as generalizing curation practices to other communities. This thesis makes an inquiry into the curation practice in the software developers’ community with the hope to shed light on the part that the GitHub context plays in curation and the roles and functions of curation in supporting the software developers’ community.

There is a large number of prior studies focusing on curation on social media. However, rarely do we find any literature that attends to curation in the software developers’ community. Therefore, the overarching goal of this dissertation is to close this gap by understanding and supporting such phenomenon.

The investigation starts with the following research questions:

*RQ1: How does GitHub support curation in the software developers’ community?*
GitHub is a social coding site designed to support software development practices, including allowing software developers to host source code repositories as well as to contribute to others’ projects. It is not designed with the mindset to support curation, which is an entirely different practice. How curation repositories are different from software repositories and how the technical features of GitHub support curation is currently unknown. The first study (Chapter 3) answers these research questions, providing insights into the GitHub features that support curation in the software developers’ community.

Next, as one of the major goals of this thesis is to find the impact of curation repositories on the software developers’ community, our investigation will then focus on the stakeholders of the curation practice, including the user experiences of both curators (Chapter 5) and users of curation repositories (Chapter 6). The following research question is therefore examined subsequently.

**RQ2: What are curators’ experiences with curation on GitHub?**

This research question focuses on understanding the experiences of the owners of curation repositories. The owners of curation repositories are software developers who create and maintain curation repositories on GitHub. We attempt to understand the motivations that drive them to curation, the technology choice, and the goals they try to achieve with curation.

Last, this dissertation also tries to understand curation practices from a user’s perspective in order to investigate how users interact with curation repositories and to learn how to improve the practice in the software developers’ community. The following question will be addressed.

**RQ3: What are software developers’ experiences with curation on GitHub from a user’s perspective?**

Users of curation repositories are software developers who visit curation repositories. As shown in Figure 1-1, curation repositories usually contain a list of indexed resources with simple descriptions and hyperlinks. The motivations that drive software developers to visit such
repositories are currently unknown. In addition, curation repositories on GitHub represent an appropriation of a technology that was designed for other purposes. It is interesting to study what are the perceived effectiveness of the current interfaces of curation repositories and how to improve it.

Figure 1-2 summarizes the research questions and issues this thesis attempts to address. Primarily, this thesis intends to understand curation practices in the software developers’ community, including how the technology supports the practice (GitHub Context) and the two types of stakeholders’ experiences (curators and users) with such practice. In addition, it also makes attempts to improve the practice with design efforts.

**Dissertation Structure**

The thesis is organized as follows. Chapter 1 describes the challenges software developers are facing in the social media era and introduces the emergence of curation on
GitHub, providing the background and motivation why we investigate the curation practice in the software developers’ community. Chapter 2 reviews the related theories and studies that this thesis will build on, including the theory of Communities of Practice, software developers’ participation in social media, and curation-related studies. Chapter 3 will discuss related literature for each research question. Chapter 4 to 6 will include three related studies on understanding and supporting curation practices. Specifically, Chapter 4 answers the first research question concerning how GitHub features support curation and the roles of curation repositories in the software developers’ community. It is accepted and published in the *Journal of Future Internet* in March 2018. Chapter 5 includes a qualitative study to understand the curators’ motivations and experiences on GitHub. The original paper is accepted and published on *PeerJ* in October 2017. Chapter 6 investigates user experiences of curation repositories on GitHub and presents a preliminary study on the design and evaluation of *RepoHunter*, which adds additional information to curated items to aid navigation inside a curation repository. The results and analysis are published in *iConference* 2016. Chapter 7 summarizes the contributions of this study and derives future directions on curation in the software developers’ community.
Chapter 2

Foundations

This section discusses the theories and prior empirical results that this thesis is built upon. Specifically, it reviews (1) the concept of communities of practice and how it affects software practices in the context of social media and (2) the concept of curation and its application in social media, laying the ground for developing our inquiry into curation in the software developers’ community.

Communities of Practice

Communities of practice are groups of people who share similar interests towards a topic and deepen their knowledge and expertise by interacting with others on an ongoing basis (Wenger et al., 2002), such as the software developers’ community. A community of practice is more than technical knowledge or skills associated with some task (Lave & Wenger, 1998). The community members are involved with a set of relationships overtime (Lave & Wenger, 1991) and communities develop around things that matter to others (Wenger, 1998).

Learning lies at the center of communities of practice (Lave, 1991). Communities of practice support continuous learning for core and peripheral members. Onboarding new members and educating existing members are essential for communities of practice to sustain and grow (Lave & Wenger, 1991; Wenger, 1998; Wenger & Snyder, 2000). Specifically, Lesser and Storck (2001) found that communities of practice can decrease the steepness of the learning curve for
new members in the organizational environment. New members of a community develop a changing understanding of practice over time from participating peripherally in ongoing activities of the community, and gradually become old-timers (Lave, 1991).

Motivations for participation and collaboration among members determine the success of communities of practice (Ardichvili et al., 2003). Communities of practice have become associated with finding, sharing, transferring, and archiving knowledge, as well as generating explicit or tacit knowledge (Becerra-Fernandez & Sabherwal, 2014). Understanding the motivational factors that lead to members’ participating in community knowledge generating and sharing activities can steer the community to more active participation (Ardichvili, 2008). In addition, knowledge is shared and transferred among members as a result of their interactions. Therefore, collaboration is another principal component in communities of practice (Ardichvili et al., 2003).

**Technology and Communities of Practice**

Technologies and communities of practice are deeply connected (Hoadly, 2012). Hoadley and Kilner (2005) identified three areas that technology can support communities of practice: content, process, and context. The content affordance refers to the ability of technology to store and manipulate information in a variety of formats, to transmit representations across distance and time, and to support human representational capacity; the process affordance refers to the ability of technology to scaffold certain tasks and activities; the context affordance refers to the ability of technology to shift the social context of the user, such as social networking sites that allow users to communicate to broader audience groups as comparing to face-to-face situations (Hoadley, 2012).
Hoadley (2012) also pointed out four techniques by which technologies can support communities of practice:

1. linking people with others who have similar practices,
2. providing a shared repository of information resources,
3. providing tools for discussing with others,
4. providing awareness in a community of the information context of various resources.

Some of these points are exemplified in the software developers’ community already, which will be discussed in the following subsections.

**The Software Developers’ Community**

Software developers need to learn constantly throughout their professional lives to meet the rapid changes of technological innovation and keep abreast of the latest knowledge and skills in software engineering (Uden and Dix, 2004), and communities of practice have been recognized as an important way for software developers to organize their learning activities. Ye and Kishida (2003) found that learning is one of the major motivational forces that attract software developers to participate in the development of open source software and become a member of the community.

It is necessary for community members to interact with each other regularly to share knowledge and collaborate in pursuit of a common class of problems (Ye and Kishida, 2003). A wide range of technologies that support the software developers’ community have been recognized and studied. And since software development does not put hard constraints on time and space, software developers often connect to each other through technologies at global scales (Storey et al., 2017).
**Early Technologies that Support the Software Developers’ Community**

Since the emergence of the Internet, software developers started to form online communities to share resources and knowledge with each other (Ye and Kishida, 2003).

Email lists have played an ongoing role in keeping members up to date with project activities. They have been used as a channel to disseminate commit logs from software repositories (Gutwin et al., 2004), supporting project awareness and coordination. They have also been used for asynchronous code reviews in open source projects by sending small patches to members for review (Rigby and Storey, 2011; Rigby et al., 2012).

Handel et al. (2002) studied a customized version of an Internet Relay Chat (IRC) chat tool and found that globally distributed developers predominantly used it for technical discussions (Handel & Herbsleb, 2002), but they also found that adoption was inconsistent across development teams (Herbsleb et al., 2002).

**Social Media and Software Developers’ Community**

The advancement of social media largely extends users’ ability to create, modify, share, and discuss Internet content (Kietzmann et al., 2011). “Social media” is the general term for a group of Internet applications that leverage Web 2.0 technologies and allow users to create and exchange content (Kaplan & Haenlein, 2009). Social media have largely reshaped the software practice in online communities (Storey et al., 2010) as well. It enables large communities of software developers to share knowledge and source code, and to create software collaboratively in a public and traceable manner (Singer et al., 2013). A survey study of software developers in the year 2010 revealed that developers used social media intensively, ranging between one and seven social media sites on a regular basis (Black et al., 2010). Commonly used social media
applications in the software developers’ community include Twitter, Stack Overflow, Hacker News, and GitHub (Storey et al., 2014; Storey et al., 2017). This subsection reviews how these social media sites support the software developers’ community.

**Twitter**

Twitter is a popular microblogging service, where users can write a short message, i.e., Tweet, and broadcast this to all his/her followers. A user can also connect to other Twitter users through following in order to get updates when followed users publish new Tweets.

Dejin and Rosson (2009) found that the technology affordances (e.g., brevity, mobility, broadcast nature) of Twitter reduces the cost of information sharing. Chatfield and Brajawidagda (2012) found that Twitter messages can reach a broad audience base in a timely manner.

The small cost of information sharing, large audience base, and speed of communication make Twitter ideal for the software developers’ community to share and disseminate information. Tian et al. (2012) found that the top five popular categories of information software developers use Twitter for to share are: (1) job openings, (2) news about a topic, (3) Q&A, (4) tools and code, and (5) tips on how to perform a task.

In addition to information sharing, software developers also leverage Twitter to stay aware of the latest industry changes, to learn through Q&A, from experts, or by conversation, and to build relationships with others (Singer et al., 2014).

**Stack Overflow**

Stack Overflow is a social Q&A website for software developers (Vasilescu et al., 2014). Users of Stack Overflow post and answer questions related to software development and can
make comments with regard to either questions or answers (Singer et al., 2013). The content is moderated by community experts, who lend themselves trustworthiness and value. In addition, Stack Overflow employs gamification features, such as reputation scores and unlocking new communities through participation, which encourages member participation (Storey et al., 2014). Of all Stack Overflow questions, 92.6% received at least one answer, and the median waiting time for the first answer and the accepted answer is 11 minutes and 21 minutes, respectively (Mamykina et al., 2011).

**Hacker News**

Hacker News is a social bookmarking site dedicated to technology-related issues and news, start-ups, and the hackers’ culture (Virasoro et al., 2011). It allows users to submit Internet links to content from all over the web (Stoddard, 2015). It is an important source for software developers to disseminate knowledge, discover new software / tools, and keep up to date with the industry (Storey et al., 2014). In addition, a software project reaching the front page of Hacker News can draw great attention from the community, where it will receive valuable feedback from the community members, which in turn helps the project grow and brings benefit to the community as a whole (Storey et al., 2014).

**GitHub**

GitHub is a crucial service for millions of software developers to share source code and collaborate with others. It is worth reviewing the recent development of research literature on GitHub to understand its technical affordances as well as its relationship to the software developers’ community.
In recent years, GitHub has drawn attention from researchers who have examined its features that promote transparency, such as activity traces, user profiles, issue trackers, source code hosting, and collaboration (Storey et al., 2014; Dabbish et al., 2013). Researchers have examined in detail how such transparency allows software developers to engage with software practices in the community (Dabbish et al., 2012; Doll, 2013; Singer et al., 2013). For example, Dabbish et al. (2012) found that the activity logs and user profiles on GitHub motivate members to contribute to software projects (Dabbish et al., 2012). Marlow et al. (2013) discovered that developers use a variety of social cues available on GitHub to form impressions of others, which in turn moderates their collaboration (Marlow et al., 2013). Singer et al. (2013) put GitHub in a larger social media environment and learned that software developers leverage transparency of socially enabled tools across many social media services for mutual assessment (Singer et al., 2013).

These studies focus on how the technology affordances of GitHub and other social media affect software practices, including learning, communication, and collaboration (Storey et al., 2014).

**Challenges Software Developers Facing in the Era of Social Media**

It is without doubt that social media sites have rapidly reshaped software developers’ online participation, bringing benefits, like being aware of the industry change, accessing expertise, and publishing software projects, that were hard to obtain in the pre-social media era. At the same time, it also introduces a number of challenges software developers are facing constantly.

Recent events have demonstrated that in the social media era, even for a single event, a huge amount of information is likely to be generated (Liu, 2008). Liu (2008) examined the social
media productivity surrounding the British Petroleum (BP) Deepwater Horizon oil spill in the Gulf of Mexico that began on April 20, 2010. At that time, over 33 million results in total with just over 2.2 million results in just the past 24 hours as of July 2010 showed up with a Google search on “BP oil spill”; in the meantime, over 500 Facebook groups, around 16 thousand Flickr photos, around 134 thousand YouTube videos, and over 1.6 million blog posts about the event were generated. The sheer volume of information generated on social media sites is becoming an issue that every user is facing.

In addition to the quantity, quality of content is another big concern in the social media era. The distribution of content quality on social media sites has high variance: from very high-quality items to very low-quality, sometimes abusive content (Agichtein et al., 2008). Bian et al. (2009) found that on a community Q&A site, the quality, accuracy, and comprehensiveness of content vary highly, and a large portion of answers do not answer users’ questions.

Software developers are facing similar challenges in the social media era. The software development industry is changing rapidly: new programming languages appear almost monthly; new programming tools (libraries, frameworks, etc.) appear almost daily; new software development methodologies appear several times a year (Jones, 2009). Storey et al. (2017) identified that software developers confront the following difficulties as a result of too many media channels to participate:

- It is difficult to keep up with new technologies and software projects.
- Too many communication channels create distractions and interruptions, negatively impacting developer productivity.
- Although there are many existing social channels, it is still difficult to find developers to participate.
• Software developers need to be literate with many communication channels. Failure to understand the channel affordances can lead to difficulties for software developers to collaborate, share, or connect with others across communities.

• The use of many channels leads to information fragmentation.

• The quantity of communicated information is overwhelming.

• The quality of communicated information is hard to evaluate. Sometimes information one finds in a media channel may already be obsoleted.

Identifying content of high quality from the large information space is becoming a real issue in the software developers’ community.

Curation

Curation is an integral element of the archaeological process and refers to the long-term management and preservation of archaeological materials and their associated documentation. Specifically, it involves activity of collecting, evaluating, organizing, and preserving a set of resources for future use (Bamforth, 1986). It was discussed at length by Binford (1973) as an archeological term (Shott, 1996), which was defined as the relationship between a tool’s potential usage and actual usage. Curation is about people adding their qualitative judgment to what is being gathered and organized (Rosenbaum, 2011). In the Internet era, technology assisting curation is commonly referred to by librarians and archivists as “digital curation” to preserve digital materials (Higgins, 2011).

“Social curation” is defined as the human process of remixing social media content for the purpose of further consumption (Duh et al., 2012). Social curation sites, such as Pinterest, offer a new way of socially creating, curating, and sharing information on the Web (Hall & Zorro, 2015).

8 https://sha.org/resources/curation-standards-guidelines/
It shares features used for social bookmarking, where users specify keywords or tags for the Internet referencing that helps organize and share curated resources with a larger community (Farooq et al., 2007).

**Curation Activity**

Curation is considered as the layer between the universe of content and the limited attention of users (Askalidis & Stoddard, 2013). In 2008, Liu (2008) realized that the large amount of information online on just one event is becoming more commonplace and is likely to increase exponentially in the near future; it is becoming easier to generate information than it is to consume it (Liu, 2008). The goal of curation is to collect content, assess the quality of each item, and promote a relatively small set of content that others should pay attention to (Askalidis & Stoddard, 2013), so that users can largely reduce the amount of information and resources they need to receive and process.

Social curation is based on the basic concept of media curation proposed by Rosembaum (2011), deals with large corpora of content from diverse sources, and connotes the activities of identifying, selecting, verifying, organizing, describing, maintaining, and preserving existing artifacts as well as integrating them into a holistic resource (Rotman et al., 2011). Social curation is considered as complementary and as a competition to information seeking behavior. Although search engines are becoming more powerful, they still lack the ability to offer the most relevant and limited number of results. Social curation complements search engines with human computing ability. The manually selecting, organizing, and evaluating behavior in the curation process extracts most relevant and useful resources according to the curator, with the intention of being useful to others and the potential for future use.
Besides its complementary nature to search behavior, social curation is also a collective process. Social curation on social media is essentially organized by social networks within the social media platforms and performed by individuals. However, individuals can curate content collectively on social curation sites (Liu, 2008; Rotman et al., 2011; Lam, 2012). It can harness the wisdom of crowds by collecting information from the wider public, allowing everyone to determine the value and veracity of the filtered content.

Moreover, social curation has the characteristics of effective distribution and consumption. Social curation is not only content selection, organization, and evaluation; it also contains the process of distribution. Sharing news on the Internet is not a new phenomenon (Hermida et al., 2012; Villi, 2012). But social media can make the sharing curated contents easier and efficient.

**Curator**

Users participating in social curation can be classified into three types: content creator, curator, and content consumer (Duh et al., 2012). Content creators are users who generate the content and post it to social media, where it is publicly available. Content is usually text messages (Tweets), photos, weblogs, movies, etc. Curators collect and evaluate the posted content, reorganize it, and add opinions, perspectives, and interests of curations to a list. Curation lists can be generated by individuals or through interaction of multiple people (Lam, 2012). Content consumers receive, share, and process the content created by content creators, as well as content expressed by curation lists (Duh et al., 2012).

Traditionally, curation was performed by experts, such as front-page editors (Askalidis and Stoddard, 2013). This is also labeled as expert curation (Macek, 2013). Such curators possess large amounts of domain knowledge, which is the reason why computers currently cannot
substitute curators (Rebholz-Schuhmann et al., 2005). Usually, the curator is a content specialist responsible for the collection of an institution, involved in the interpretation of heritage material. Being responsible for the collection, he has the duty to preserve but also to enhance its value and to share its content to the public.

Social curators are knowledge brokers that interpret, publicize and endorse content (Villi, 2012). According to Macek (2013), everyone rather than just domain experts can participate in social curation. The number of media artifacts and content sources is fast growing, shadowing traditional expert curators. The role of a social curator resembles the classical role of curators, such as museum curators (Rosembaum, 2011): a curator on social media has almost the same tasks. In particular, the curation of professional and social media content, such as aggregating, selecting, organizing (Rotman et al., 2011), and presenting news according to the criteria for high quality journalism from professional and user-generated content, results in new types of editorial content and experience for users.

**Curation in Social Media**

Digital curation usually refers to any curation activities that concern digital information. For example, Beagrie (2008) defined digital curation as “... is about maintaining and adding value to a trusted body of digital information for current and future use”. There is also literature that limits digital curations to academic purposes. Curry et al. (2010) referred to digital curation as “... the process of establishing and maintaining a trusted body of digital information within long term repositories for current and future use by researchers, scientists, historians, and scholars generally”.

Later, as user-generated content became popular, content curation emerged, referring to users categorizing and organizing collections of content created by others that they found online,
providing an editorial perspective by highlighting interesting content (Zhong et al., 2013). In recent years, a number of content curation services, such as Pinterest, Tumblr, and Scoop.it, emerged and received wide attention from the public. These applications allow users to curate collections of content, such as images, news, and web pages, of their own interest and share with others (Saaya et al., 2013).

Social media applications encourage users to create and share different content with others online. Researchers are starting to think about how social media, over time and across sites, form part of the wider digital archiving space for individuals (Zhao & Lindley, 2014). Social curation especially refers to the curation behaviors that are found on social media. It is sometimes used interchangeably with content curation (Zhong et al., 2013), but more often, social curation refers to the curation behaviors on social media sites. For instances, Stanoevska-Slabeva et al. (2012) specifically used social media curation to refer to the curation behavior on social media sites as compare to content curation: “the result of social media curation [is] curated news containing selected original contributions from social media that are glued together to craft stories with context and background information provided by the curator, i.e., the author of the story”; Duh et al. (2012) simply defined social curation as the curation of any social media content. Some researchers, among whom Zhong et al’(2013), use the term “social curation” because of the social components associated with social media sites: “users can follow other content curators that they find interesting, as a way of gaining exposure to new and interesting content” (Zhong et al., 2013). The distinguished social features of social media, such as knowing what others are interested in, sharing with the public, and getting feedback from the public, important characteristics of social curation, because those features and functions are not included, or at least not emphasized, in digital curation or data curation.
Current Curation Services

Ishiguro et al. (2012) listed a set of features curation services should offer to users, which include:

a) bundling a collection of content from diverse sources,
b) reorganizing them to give one’s own perspective,
c) publishing the result to consumers.

Many highly popular curation services incorporated those features. For instance, delicio.us allows users to categorize interesting URLs by tagging them and sharing them with followers. Digg.com and reddit.com allow the sharing of news articles or other types of stories. Pinterest is considered the most popular content curation website for sharing pictures and videos, and Last.fm is a popular social music curation service (Zhong et al., 2013). Gilbert (2013) provided a detailed study of empirical phenomena on Reddit and observed that a large portion of popular articles was actually submitted to Reddit multiple times before the content became popular.

Services like Twitter, Pinterest, and Reddit are services for the general population with diverse backgrounds and interests, while GitHub is intended for a focused community of software developers. Members of the software developers’ community share a set of common goals and practices, which is likely to affect their participation in curation practices as well. Second, unlike Pinterest, which is designed for curation of links, GitHub is an online work platform designed for software developers to collaborate with others on software projects, and curation is an appropriation of the collaborative coding features of the platform. The reasons behind such appropriation and whether GitHub features fully meet curation needs of developers are yet to be discovered. Third, the technology affordances of GitHub largely depart from the abovementioned services. Tools like Pinterest and Flickr are designed for personal collection and sharing of
hyperlinks. Reddit allows users to vote to promote links, but it hardly preserves resources.

GitHub provides a collaborative working space, i.e., the repository, where software developers can work on the same project together and are enforced by Git workflow. Therefore, GitHub is distinct regarding user base, intended purpose, as well as technology affordances. Its appropriation for curation purposes raises an interesting question concerning users’ motivations and experiences.

**Curation Repositories on GitHub**

---

**Continuous Integration**

*Tools for help with continuous integration*

- drone - Drone is a Continuous integration platform built on Docker, written in Go
- goveralls - Go integration for Coveralls.io continuous code coverage tracking system.
- overalls - Multi-Package go project coverprofile for tools like goveralls

**CSS Preprocessors**

*Libraries for preprocessing CSS files*

- c6 - High performance SASS compatible-implementation compiler written in Go
- gcss - Pure Go CSS Preprocessor.
- go-libsass - Go wrapper to the 100% Sass compatible libsass project.

**Data Structures**

*Generic datastructures and algorithms in Go.*

- binpacker - Binary packer and unpacker helps user build custom binary stream.
- bitset - Go package implementing bitsets.
- bloom - Bloom filters implemented in Go.

---

Figure 2-1. A part of the README.md file of the awesome-go⁹ curation project.

Curation practices are enabled by GitHub features. Specifically, GitHub introduces a README.md file in the root directory of each repository. The content of the README.md file is displayed on the front page of the repository, i.e., if a user visits the URL of a software

---

⁹ [https://github.com/avelino/awesome-go](https://github.com/avelino/awesome-go)
repository hosted on GitHub in a browser, the README.md file will be displayed as a web page (Figure 2-1) along with the repository structure and some project statistics, such as the number of forks and stars (McDonald & Goggins, 2013). The content of the README.md file can be structured with Markdown syntax\(^\text{10}\), which provides rich text features, including a table of contents, links, and tables. README.md is designed for adding descriptions and documentation for a repository\(^\text{11}\).

Curation on GitHub appropriates the README.md file of a repository to create a list of resource indexes within one page. It categorizes resources into different themes and differentiates them into sections. Typically, each resource is recorded with the resource name and a brief description of the resource (see Figure 2-1). In addition, URLs are attached to each of the curated items. Clicking a resource name (shown in blue in Figure 2-1) will direct the user to the real web location of the resource.

**Gaps**

Communities of practice are prevalent in software engineering, where software developers constantly improve their programming skills by ongoing participation in the community activities (Storey et al., 2014). Technologies, such as mailing lists and IRC channels (Handel et al., 2002; Gutwin et al., 2004), have a long history of supporting the software developers’ community. In recent years, software developers have adopted a varied set of social media applications to organize their communications in the communities of practice. These social media sites bring abounding benefits to the community, such as reducing the communication cost, reaching a broad audience base, and making the software development process more transparent.

\(^\text{10}\) [https://help.github.com/articles/basic-writing-and-formatting-syntax](https://help.github.com/articles/basic-writing-and-formatting-syntax)
\(^\text{11}\) [https://help.github.com/articles/create-a-repo](https://help.github.com/articles/create-a-repo)
and traceable. At the same time, they also introduce ample challenges for software developers, such as information quality, fragmentation, and information overload.

Social curation, which is the process of manually identifying, selecting, organizing, and evaluating information and resources generated by others, and sharing the result to the public is likely to mitigate the problem. Social curation can filter out noises, generate resources of high quality, and largely reduce the number of pieces of information or resources that content consumers need to pay attention to.

The curation on GitHub is a type of social curation, where software developers collaboratively appropriate the GitHub repository to filter and maintain repositories of resources for future use. We found no prior literature on the investigation of the context and technology affordances for such practice. Therefore, this thesis work will first attempt to understand how GitHub features and environment support the curation practice in the software developers’ community.

Second, although we found by reviewing the related literature that curation may address the large quantity of information generated in a community as well as the large variance of the quality of the information, we currently have little understanding of the motivations that drive software developers to curate resources, especially to appropriate GitHub for such purpose. Therefore, we will investigate the motivations and the experiences of curators.

Third, how users interact with curation services, such as Pinterest, well documented. However, why and how users interact with curation repositories on GitHub is still unknown. Our last research effort will be devoted to understanding and improving users’ experiences with curation repositories.
Chapter 3

Related Studies

The previous chapter discussed the foundations of curation in the software developers’ community, including (1) the roles of technologies in the software developers’ community as communities of practice, (2) the challenges of software developers participating in a variety of social media, and (3) curation activities and current curation services.

This chapter will focus on reviewing the related literature for each research question we intend to answer. It is organized as follows. First it introduces curation repositories in the GitHub environment, which includes the common GitHub technical affordances. Then it reviews the related studies that are related to each respective research question.

Curation in the GitHub Environment

Curation repositories appropriate the GitHub repository to select, evaluate, organize, and preserve a set of resources. This subsection presents the GitHub features that enable curation in the software developers’ community.

GitHub Repository

Activities on GitHub are organized around repositories. Specifically, a GitHub repository is the basic unit to organize a software project and related activities\(^\text{12}\). Each repository contains pages such as an issue tracker, wiki, and the pull request. GitHub users can send pull requests to a

\(^{12}\) [https://guides.github.com/activities/hello-world](https://guides.github.com/activities/hello-world)
repository for making code contributions, comment in the issue tracker for bug reporting or feature requesting or edit the wiki pages for documenting the project (Dabbish et al., 2012; Wu et al., 2014).

In addition to the software development-related features listed above, GitHub provides a set of UI components for users to interact with a repository. A GitHub user can “watch” a repository in order to receive the updates from the repository. In addition, in 2012, GitHub announced a new feature that allows users to mark a repository as interesting by giving it a “star”\(^{13}\). Today, stars are considered an important indicator of the popularity of a repository (McDonald & Goggins, 2013).

A curation repository is a GitHub repository that is created for the purpose of curation, which is an appropriation of GitHub features designed for software collaboration. We are interested in understanding (1) the way in which GitHub features are adopted and (2) how activities are organized around a curation repository.

**RQ1: How Does the GitHub Context Support Curation?**

This research question intends to understand how GitHub environment supports curation repositories.

**The Purposes of Curation in GitHub**

The purposes of curation have been a popular topic in curation-related literature. Studies of the purposes of curation in the social media environment are well documented. For example, Duh et al. (2012) found entertainment and hobbies and serious topics, such as society, politics,

\(^{13}\) [https://github.com/blog/1204-notifications-stars](https://github.com/blog/1204-notifications-stars)
and economics, were among the most favored topics when curating Tweets. Change et al. (2014) discovered that the most popular categories of pins are “food and drink”, “DIY crafts”, “home decoration”, and “women’s fashion”.

However, curation in the GitHub environment is different from that on social media sites, such as Twitter and Pinterest, in the following ways. First, the user base of GitHub is converged, where the dominating population is software developer, focusing on software practices. Second, GitHub concentrates on software practice, where features for collaboration are emphasized (Dabbish et al., 2012; Marlow et al., 2013; Tsay et al., 2014), while curation in other social media sites focused on individuals and does not emphasize collaboration among its users (Zhong et al., 2013; Chang et al., 2014). The comparison leaves an interesting question concerning the goals of curation in GitHub as an entirely different environment.

**The Curation Leadership and Content**

Matthews et al. (2014) reported that in a large enterprise environment, curators are usually the community leaders. They found that Wiki is usually the tool an intranet community used to map internal and external content, and the knowledge is usually created and maintained by a small set of people in the leadership team, rather than through community-wide efforts (Matthews et al., 2014). We wonder if the same is true in the GitHub environment, i.e., whether curation is carried out mostly by a community leader or a group of people from the leadership team.

The number of followers an individual software developer has is an indicator of community status in GitHub (Dabbish et al., 2012; Marlow et al., 2013; Tsay et al., 2014). An example of a “coding rock star” is GitHub user “dhh” (Dabbish et al., 2012), who has thousands of followers. Prior work has found that the distributions of the number of followers shows a
power law-like shape (Lima et al., 2014). Therefore, whether a GitHub user is a community leader can be indicated by the number of followers s/he has. In the analysis of this part, we are curious if the owners of the curation repositories are leaders in the GitHub environment, i.e., GitHub users with a large number of followers.

Data provenance of curated content is a related topic. Matthews et al. (2014) captured the source of each curated item in an organizational environment and found that both internal and external resources are curated with respect to different tools (wiki, blog, etc.). The curation efforts in intranet communities are often performed by community leaders, while members of communities are encouraged to curate external resources (Matthews et al., 2014). We would like to follow a similar procedure to analyze the sources of curated items and the status of the owners of curation repositories on GitHub.

**The Collaboration Patterns in Curation Repositories**

Last but not least, since GitHub is a highly collaborative environment, we also wonder whether and how collaborative features, especially pull requests, are used in curation repositories. Software developers on GitHub can get involved with a repository by forking a repository, making changes, and then issuing a pull request to have their change merged back into the main branch of the project (Marlow et al., 2013). Forking and pull requests create a development model where the changes are pushed to the owner of the repository and go through code review by the community before being integrated (Kalliamvakou et al., 2014). We wonder if community members go through the pull request process to review the curated item before they are merged to a curation repository.

Moreover, as pull requests show the changes of a repository, the ensemble of all pull requests of repositories reveal how a repository is changing over time. The analysis of how
curation repositories are changing over time through the lenses of pull request is currently uninvestigated. This study also intends to close this gap by examining the types of pull requests sent to curation repositories over time.

**RQ2: What Motivates Software Developers to Curate on GitHub?**

This section reviews the previous literature that explores tools for curation, individuals’ motivation to curate, and software developers’ motivations in participating in online communities.

**Motivations to curate in the social media era**

Curation is a common practice in archeology. It is the activity of collecting, evaluating, organizing, and preserving a set of resources for future use (Bamforth, 1986). In the Internet era, technology-assisted curation to preserve digital materials is commonly referred to by librarians and archivists as “digital curation” (Higgins, 2011). It can share features used for social bookmarking, where users specify keywords or tags for the Internet referencing that helps organize and share curated resources with a larger community (Farooq et al., 2007). There are several early popular social bookmarking tools, such as del.icio.us, which allows sharing of personal bookmarks (Golder & Huberman, 2006), Flickr, a photo tagging and sharing service (Marlow et al., 2006), and Reddit, a community-driven link sharing, commenting, and rating service (Singer et al., 2014). Curation behaviors have been further studied since social media were appropriated to enable new forms of curation. Specifically, Duh et al. (2012) reported the use of a third party tool, Togetter, for curating Tweets, and uncovered the intended purposes for these curated lists, including recording a conversation, writing a long article, and summarizing
an event (Duh et al., 2012). Zhong et al. (2013) conducted surveys among Pinterest and Last.fm users and found that the majority of the users engage with the curation site for personal interests rather than social reasons (Zhong et al., 2013). A recent study examined the ways that communities leverage a variety of social tools for curation to support vital community activities in a large enterprise environment (Matthews et al., 2014). The authors also call for future studies on curation in public Internet communities (Matthews et al., 2014).

Curation on GitHub is a unique instance of curation set apart from the above studies in the following ways. First, the user body of GitHub is drastically different. Services like Twitter, Pinterest, and Reddit are services for the general population with diverse backgrounds and interests, while GitHub is intended for a focused community of software developers. Members of the software developers’ community share a set of common goals and practices, which is likely to affect their participation in curation practices as well. Second, unlike Pinterest, which itself is designed for the curation of links, GitHub is an online work platform designed for software developers to collaborate with others on software projects, and curation is an appropriation of the collaborative coding features of the platform. The reasons behind such appropriation and whether GitHub features fully meet the curation needs of developers are yet to be discovered. Third, the technology affordances of GitHub largely depart from the abovementioned services. Tools like Pinterest and Flickr are designed for personal collection and sharing of hyperlinks. Reddit allows users to vote to promote links, but it hardly preserves resources. GitHub provides a collaborative working space, i.e., the repository, where software developers can work on the same project together and are enforced by the Git workflow. Therefore, GitHub is distinct regarding user base, intended purpose, and technology affordances. Its appropriation for curation purposes raises an interesting question concerning users’ motivations and experiences.
Software developers’ motivations for participating online communities

Researchers report two main categories of motivations that drive software developers’ voluntary participation in open source software projects: (1) internal motivations, i.e., intrinsic motivations, altruism, and community identification, and (2) external rewards, including expected future rewards and personal needs (Hars & Ou, 2001; Ye & Kishida, 2003). Internal factors include “intrinsic motivation”, which refers to the feeling of competence, satisfaction, and fulfillment as a motivator to participate in open source projects; “altruism” refers to software developers’ desire to care for others’ welfare at their own cost; and “community identification” refers to individual software developers’ alignment of goals with the larger community. External factors include “future rewards” when software developers view their participation as an investment and expect future returns, including revenues from related products and services, human capital, self-marketing, and peer recognition; “personal needs” are software developers’ personal demands for their activity; for example, the Perl programming language and Apache web server both grew out of software developers’ self-interests to support their work (Hars & Ou, 2001). Both internal and external factors are important motivations that drive software developers’ participation in open source projects.

The rise of social media impacts the way software developers participate in online space. Social media are often referred to as socially enabled tools, where social features are added to software engineering tools (Storey et al., 2014). It lowers the barrier to publishing information, allows for fast diffusion, and enables communication at a large scale, which facilitates a “participatory culture” in the software developers’ community (Storey et al., 2014; Jenkins et al., 2009). As a result, software developers increasingly participate in the social media community, which enhances learning, communication, and collaboration (Dabbish et al., 2012; Doll, 2013; Singer et al., 2013). Similarly, software developers are motivated to participate in order to
satisfy personal needs (e.g., improve technical skills) and to gain peer recognition (e.g., recognition by the community) (Storey et al., 2014).

Despite the well-studied motivations for software developers’ participation in online communities, software developers’ motivation to engage in curation practices within GitHub by appropriating a collaboration on software development features are currently under-explored.

**RQ3: What Are the Software Developers’ Experiences with Curation from a User’s Perspective**

Software developers visit curation repositories on GitHub to satisfy their information needs. Questions remain on what their information needs are, what types of information behavior they exhibit in visiting curation repositories, and how well GitHub features support their visits of curation repositories. We use the Information Discovery framework developed by Voyloshnikova and Storey (2014) to analyze software developers’ information behavior in curation repositories.

**Information Discovery Framework**

Voyloshnikova and Storey (2014) propose an information discovery framework that contains four types of information seeking behaviors: serendipitous discovery, fact discovery, rediscovery, and channel-based discovery.

**Serendipitous discovery**

Serendipitous discovery is a kind of information seeking behavior that encounters an unexpected source, miscellaneous fact, or familiar situation that may be of some use in meeting current or future needs (McKenzie, 2003). This kind of information discovery is characterized by under-defined, absent, or hidden information needs, and it usually involves browsing through diverse resources with varying content types (Keller et al., 2006; Voyloshnikova & Storey, 2014).
A resource is defined as a collection of information about a single unit of inquiry, usually bundled together for presentation purposes, e.g., places, images, blog posts, or Web pages (Voyloshnikova & Storey, 2014).

**Fact Discovery**

Fact discovery refers to information discovery resulting from the search for a specific piece of information. Contrasting to serendipitous discovery, it is usually characterized by a well-defined information need and is easier to perform within systems that provide access to homogeneous types of information (Voyloshnikova & Storey, 2014). Since serendipitous discovery and fact discovery are comparable, we wonder (1) whether users of curation repositories perform these two types of information discovery behavior, (2) what is the proportion of each type of information discovery behavior, and (3) whether these two types of information discovery behavior are effective.

**Rediscovery**

Rediscovery refers to information discovery resulting from revisiting previously discovered resources (Voyloshnikova an Storey, 2014). Common rediscovery behavior includes history-based rediscovery, where the browsing history of a user is recorded, and the user can rediscover formerly visited pages, and bookmark-based rediscovery, where a user bookmarks a visited resource for future revisit. The GitHub “star” function performs the bookmark function. When a GitHub user “stars” a repository, GitHub will record the repository to the user’s own space, and the user can check all his/her starred repositories in the “Stars” page in his/her GitHub space. We are curious whether and how a user performs rediscovery with curation repositories.

**Channel-Based Discovery**

Channel-based discovery is enabled by features that suggest information to users based on their subscription. It includes two different information seeking tasks, monitoring and

---

awareness (Voyloshnikova & Storey, 2014). If a user actively looks for updates, then an application provides monitoring, and if a user receives notifications about updates, then an application facilitates awareness (Voyloshnikova & Storey, 2014). GitHub provides a “watch” function, which allows users to subscribe to a repository to receive the updates. We wonder whether and how channel-based discovery is involved in curation repositories.

**Motivations in Information Discovery**

In addition to the different types of information discovery behavior, another important aspect in information seeking is people’s motivation in performing the activity. People often engage in information seeking activities to close a knowledge gap that occurred as a result of not having enough information to perform a task (Proper & Bruza, 1999). Voyloshnikova and Storey (2014) argue that when providing tool support for various information discovery tasks, it is useful to consider the motivation behind these tasks as it can be different for each task.

**Information Cues**

In an information environment, an information cue is a signal that can influence users’ evaluation of information sources (Pirolli & Fu 2003). For example, in a news browsing context, the news source, the time elapsed since the story, and the number of articles that refer to the news are information cues that can influence users’ perception of a news item (Sundar et al., 2007). The transparency of GitHub provides abundant information cues. For instances, the number of followers signals whether a user is a “coding rock star”, and the number of code commits suggests the activeness of a repository (Dabbish et al., 2012). Existing research suggests these
cues lead users to form attitudes and make evaluations about information on the site (Kim & Sundar, 2011).

In recent years, GitHub has drawn great attention from researchers, who have examined its features that promote transparency, such as activity traces, user profiles, issue trackers, source code hosting, and collaboration (Storey et al., 2014; Dabbish et al., 2013). Researchers have examined in detail how such transparency allows software developers to engage with software practices in the community (Dabbish et al., 2012; Doll, 2013; Singer et al., 2013). For example, Dabbish et al. (2012) found that the activity logs and user profiles on GitHub motivate members to contribute to software projects (Dabbish et al., 2012). Marlow et al. (2013) discovered that developers use a variety of social cues available on GitHub to form impressions of others, which in turn moderates their collaboration (Marlow et al., 2013). Singer et al. (2013) put GitHub in a larger social media environment and learned that software developers leverage transparency of socially enabled tools across many social media services for mutual assessment (Singer et al., 2013).

In recent GitHub-related studies, many researchers have pointed out the transparent environment GitHub provides (Dabbish et al., 2012; Marlow et al., 2013; Singer et al., 2014). In social media-related research, transparency refers to the visibility of others’ actions on public or shared artifacts (Dabbish et al., 2012). It also includes social signals, such as the number of votes associated with an answer or the number of followers a user has (Storey et al., 2014). On GitHub, transparency, such as the number of stars and forks of a GitHub repository (McDonald & Goggins, 2013) and the visibility of developer activities and profiles, is crucial, because it can motivate others to contribute and thus shape the success of a repository (Dabbish et al., 2012; Wu et al., 2014). For example, software developers evaluate the frequency of contributions and the involvement of a “coding rock star” of a repository to decide whether a repository is worth
contributing to (Dabbish et al., 2012). Therefore, the transparent environment provides abundant information cues that are adopted by software developers to make decisions.

In answering RQ3, we ponder on the motivations that drive them to access curation repositories and the types of information discovery that apply to them. Given the prevalence of information cues on GitHub, it raises interesting questions about what information cues software developers look for when browsing curation repositories and about how the cues are useful.
Chapter 4

How GitHub Supports Curation Repositories

We start the investigation of curation in the software developers’ community by examining how curation is supported in GitHub. Recently, curation has been investigated in the context of social networking sites, such as Twitter and Pinterest, and with respect to media content, such as videos, images, text (tweets), and links to other online content and resources (pins) (Duh et al., 2012; Zhong et al., 2013). The context of curation in GitHub is distinct from the prior-mentioned curation services in two ways. First, curated repositories in GitHub are directed at the software developers’ community, whose members share and cultivate a professional interest in software development. Second, curation practice on GitHub is embedded in the context of the software developers’ community, which organizes and coordinates software practice with an ensemble of social coding tools not intended for curation. This context raises interesting questions concerning how GitHub features support such an appropriation for curation. Thus, this study addresses the following research question:

*RQ1: How does GitHub, a social coding service, support curation in software developers’ community?*

To answer this research question, we first compare how curation repositories are different from typical software repositories. As a relatively new way to utilize GitHub, users are likely to participate in such kind of repository differently. This relates not only to the number of GitHub users who star a repository, which usually shows a user’s interests towards a repository (McDonald and Goggins, 2013; Tsay et al., 2014), but also the different types of activities that
take place inside a curation repository, such as pull requests (Tsay et al., 2014). Thus, we explore our initial research question through the specific sub-research questions outlined below.

**RQ1.1: How are curation repositories different from the typical software repositories of GitHub?**

Prior literature has documented GitHub software repository practices well (Dabbish et al., 2012; Marlow et al., 2013; Tsay et al., 2014). However, a gap exists in how GitHub features are utilized in curation repositories and how they are adopted differently as comparing to the intended software practice. This research question intends to close this gap, and to provide an account of the categories and user participation that make curation repositories different.

In addition to the comparison with software repositories, currently, we have little understanding of the details of curation repositories in terms of what needs they address and what role they play in the software developers’ community. Thus, the following research question will be investigated next.

**RQ1.2: What is the emerging role of curation repositories in the GitHub community?**

Specifically, this research question examines the function of curation by examining the contents, format, the owner’s characteristics, and collaboration pattern of curation repositories on GitHub. The answer to this research question can elucidate why curation repositories are useful, why they have suddenly drawn great attention, and what kind of impact it brings to the software developers’ community.

Through a statistical analysis of the activity logs to compare curation repositories with software repositories and a content analysis of the most popular curation repositories on GitHub, we find that curation repositories are more popular than software repositories, and GitHub users participate in curation repositories in a qualitatively different way. Most curation repositories are maintained by individual software developers, and they intend to collect and preserve high-quality resources, originated from either inside or outside of GitHub, about the technology
industry. Our findings suggest that curation repositories become an essential way for the software developers’ community to centralize fragmented information and share knowledge. This study contributes to the understanding of curation in GitHub, and sheds light on the potential ways to better support the practice.

**Method**

To characterize popular curation repositories hosted on GitHub, we collected a data set of activity logs on GitHub, identified the top curation and software repositories, compared them with top software repositories, and coded the contents of curation repositories. The dataset was collected from GitHub Archive\(^{15}\), which captures a comprehensive GitHub timeline data. GitHub Archive data has been actively used for analysis in academic publications (Lima et al., 2014; Kalliamvakou et al., 2014; Wu et al., 2014; Wu et al., 2015). However, the data set was influenced by a bug report about a crawler issue on 9/22/2013\(^{16}\), which resulted in a loss of events. In a consideration of consistency and data quality, we collected 109,782,635 events on 7,079,847 repositories that occurred between 10/1/2013 and 8/31/2014.

In order to find the commonalities of the curation repositories that are of interest to others, we selected the ones based on indicators of popularity. Trending repositories are displayed on GitHub by day, week, or month. The trending repositories typically average about 500 stars per repository\(^{17}\). Given that a repository that trends can be considered a relatively popular repository on GitHub, we then selected all repositories that had more than 500 stars within the date range. At the same time, many software repositories with more than 500 stars had been established for years. In order to have a fair comparison of curation and software repositories,

---

15 https://www.githubarchive.org/
16 https://github.com/igrigorik/githubarchive.org/pull/37
17 https://github.com/trending?l=all&since=weekly
only repositories that were created after January the first, 2013 were retained, resulting in 1,929 repositories.

To identify curation repositories within this sample, we first identified 1,384 software projects, whose programming languages are automatically detected by GitHub. For the rest 545 repositories, we manually labeled each. The criterion used to determine if a repository was a curation repository was whether the primary content of the repository was a collection of the Internet resources. As a result, we identified 49 curation repositories from the 545 repositories. As we cannot verify the nature of the other 496 repositories, and also our immediate interest in this paper is focusing on popular curation repositories, as well as comparing them with software repositories, we discarded the 496 repositories from the sample.

After identifying the most popular curation and software repositories, to answer RQ1, *how are curation repositories different from the typical software repositories of GitHub*, we aggregated the activity log data into the 49 curation repositories and the 1,384 software repositories respectively and applied a quantitative method to compare them. Specifically, for each type of activity, we analyzed whether the number of the activities for curation projects is different from the number of activities for software repositories. To answer RQ2, *what is the emerging role of curation repositories in the GitHub community*, we performed content analysis on the 49 repositories. For each curation repository, we coded the repository name, description, curated items, pull requests, and owners’ profiles, which were retrieved from GitHub on Sept. 1st, 2014. Open coding strategies as suggested by Strauss (1987) were applied to developing a coding schema (Strauss, 1987). Themes and concepts were identified, discussed, and refined iteratively among researchers (Lacey and Luff, 2001). The results are presented the following section.
Results

Curation Repository vs. Software Repository

This subsection presents the results of the quantitative analysis of the comparison between curation repositories with software repositories with regards to popularity and activities. It yields insights into how the GitHub infrastructure has been used for curation.

Figure 4-1. Average number of stars received by curation repositories and software repositories.

Popularity

We first compare the popularity of curation repositories with software repositories using the number of stars, which is an indicator of the status and popularity of a repository on GitHub (Tsay et al., 2014).
Of the top 1,433 repositories that have more than 500 stars in our sample, 49 are curation repositories and 1,384 are software repositories. The top 3 most-starred repositories are all curation repositories, and 6 out of the top 20 most-starred repositories are curation repositories.

To compare the two groups, an independent samples t-test was performed to compare the log-transformed mean of the number of stars between curation and software repositories. The reason for log-transformation is because the number of stars that a repository has does not follow the normal distribution. Rather, it is a type of count data. In software engineering log transformation is an established practice for the analysis of count data (Mockus et al., 2013). The results show that curation repositories received a significantly higher number of stars ($M=7.51$, $SD=0.99$) than software repositories ($M=6.98$, $SD=0.65$), $t(49.48)=-3.70$, $p < 0.001$. Figure 4-1 shows the average number of stars (not log-transformed) received by curation repositories and software repositories.

### Comparison of User Participation for Curation and Software

In addition to the comparison of popularity, in order to characterize how GitHub features are leveraged in curation repositories, we then compared the curation repositories and software repositories in terms of different types of user activities.

GitHub records several types of activity logs\(^\text{18}\). The important and representative activities include the following: *Create* and *Delete* events indicate a change in the repository structure, such as creating or deleting a Git branch. *Issue Comment* and *Issues* events indicate user participation in issue tracker. *Fork* event happens when a GitHub user clones a repository to his/her own space, which is a prerequisite for a *Pull Request*. A *Pull Request* event indicates a GitHub user makes some code changes of a repository and wants the owners of the repository to

merge the changes. It is an indicator of contributions from others. A Push event refers to the event of committing a code change in a repository.

The number of activities performed in each repository in our sample was aggregated and log-transformed (Mockus et al., 2013). Then independent samples t-tests were performed on each type of activity introduced above to compare curation repositories with software repositories. Since 8 t-tests were performed among the two groups (including comparing of numbers of stars for popularity), a Bonferroni-Holm (1979) correction for multiple testing was applied to avoid the issue of multiple comparisons (Holm, 1979). The results are shown in Table 4-1. All Bonferroni-Holm corrected p values are significant. Therefore, curation repositories are significantly different from software repositories for all seven types of activity. Specifically, curation repositories have higher numbers of Fork, Push, and Pull Request events, and lower number of Create, Delete, Issue Comment, and Issues events.

Table 4-1. Independent sample t-tests that compare the log-transformed mean of the number of seven types of activities between curation projects and software projects.

<table>
<thead>
<tr>
<th>Type of Event</th>
<th>Curation (N=49)</th>
<th>Software (N=1384)</th>
<th>df</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Mean (log)</td>
<td>SD</td>
<td>Mean</td>
</tr>
<tr>
<td>Create</td>
<td>1.84</td>
<td>0.80</td>
<td>0.62</td>
<td>16.97</td>
</tr>
<tr>
<td>Delete</td>
<td>0.65</td>
<td>0.21</td>
<td>0.57</td>
<td>9.40</td>
</tr>
<tr>
<td>Fork</td>
<td>465.69</td>
<td>5.45</td>
<td>1.08</td>
<td>161.81</td>
</tr>
<tr>
<td>Issue Comment</td>
<td>105.10</td>
<td>4.02</td>
<td>1.28</td>
<td>406.91</td>
</tr>
<tr>
<td>Issues</td>
<td>27.16</td>
<td>2.62</td>
<td>1.27</td>
<td>140.64</td>
</tr>
<tr>
<td>Pull Request</td>
<td>151.69</td>
<td>4.29</td>
<td>1.28</td>
<td>91.52</td>
</tr>
<tr>
<td>Push</td>
<td>405.82</td>
<td>4.57</td>
<td>1.07</td>
<td>165.36</td>
</tr>
</tbody>
</table>

* indicates the result is significant at p < 0.01, ** indicates the result is significant at p < 0.001

(log) indicates the statistics are calculated on log-transformed value.
The Emerging Role of Curation Repositories

In spite of the enormous amount of popularity curation received, the kind of information needs it addresses and the role it plays in the software developers’ community are currently unknown. In this section, we applied content analysis to examine the curated contents, the owners, the contributors, and the interactions among owners and contributors in order to understand the roles of curation repositories on GitHub.

The Purposes of Curation Repositories

First, we investigate the contents of curation repositories. As shown in Figure 1, the content of a curation repository is usually a list of curated items of various types, grouped into a set of categories. We coded the topic of each curation repository, the types of curated items and examined the provenance of the curated items.

Overall, 47 of the 49 curation repositories include resources about a topic in the technology industry, one is about curating taco recipes, and one is about curating images.

The quality indicators of curated contents

We observed that the curation repositories usually include quality indicators in the repository name, description, or README.md file. Out of 49 projects, 27 contain quality indicators, including words or phrases like “awesome” (15 repo), “must-watch” or “must-read” (3 repo), “good” (3 repo), “useful” (2 repo), and “rock your world”, “most influential”, “favorite”, “cool” once, respectively. Some curation repositories also include a very specific definition of the quality of their contents, such as:

“An awesome package is one that is mature (not recently released), is well maintained, has a good amount of users, has good documentation, follows the best practices, and which latest
release is less than 1 year old. Awesome Django packages and projects are the ones that inspire
and serve as examples.” – awesome-django

**Types of curated items**

Then we followed the hyperlinks of each curation repository and recorded the type of resource it directed to. The most common types of curated resources are articles and software projects. The complete types of resources and the associated number of curation repositories that include that type of resources is presented in Table 4-2.

<table>
<thead>
<tr>
<th>Type of resources</th>
<th>#</th>
<th>Type of resources</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Articles</td>
<td>31</td>
<td>Courses</td>
<td>6</td>
</tr>
<tr>
<td>Software projects</td>
<td>30</td>
<td>Conferences</td>
<td>4</td>
</tr>
<tr>
<td>Websites</td>
<td>24</td>
<td>Audio</td>
<td>4</td>
</tr>
<tr>
<td>Books</td>
<td>20</td>
<td>Microblogs</td>
<td>4</td>
</tr>
<tr>
<td>Video</td>
<td>12</td>
<td>Q&amp;As</td>
<td>4</td>
</tr>
<tr>
<td>Software</td>
<td>11</td>
<td>People</td>
<td>3</td>
</tr>
<tr>
<td>Blogs</td>
<td>9</td>
<td>Research Papers</td>
<td>3</td>
</tr>
</tbody>
</table>

**Curated item:**

Following Matthews et al.’s (2014) framework for identifying the source of curated content, we examined the source of curated items in each curation repository. We coded the repositories using the following scheme: repositories with more than 75% contents from GitHub were coded as internal curation, repositories with less than 25% contents from GitHub as external curation, and the rest is coded as hybrid curation.

Around 30% of the curation repositories engage in internal curation where the contents mostly come from within GitHub. In 61.2% of the curation repositories, the majority of the curated items come from external resources, which is originated outside of the GitHub realm (Table 4-3).

---

19[https://github.com/robario/awesome-django]
Table 4-3. Sources of curation repositories.

<table>
<thead>
<tr>
<th>Curation type</th>
<th># Curation repositories</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal-based Curation</td>
<td>15</td>
<td>30.6%</td>
</tr>
<tr>
<td>Hybrid Curation</td>
<td>4</td>
<td>8.2%</td>
</tr>
<tr>
<td>External-based Curation</td>
<td>30</td>
<td>61.2%</td>
</tr>
</tbody>
</table>

As seen in Figure 1, a section contains a themed list of items, which notes the resource name, link, and description. We further coded item description as follows: 1) if the curated list contains nothing except the name of the resources and hyperlink, it is coded as “No description”; 2) if most items (more than 70 percent) of the list have a one or two sentence description, it is coded as “Simple description”; 3) if most items (more than 70 percent) of the list have a the description, and the description is either structured or longer than two sentences, it was coded as “Rich description” (Table 4-4).

Around 60% of the curation repositories have some descriptions for each curated item, and 40% have no description at all. It shows the varying of formatting for curated lists, and lacking standardization.

Table 4-4. Levels of description in curation repositories.

<table>
<thead>
<tr>
<th>Level of description</th>
<th># Curation repositories</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>No description</td>
<td>19</td>
<td>38.8%</td>
</tr>
<tr>
<td>Some description</td>
<td>20</td>
<td>40.8%</td>
</tr>
<tr>
<td>Rich description</td>
<td>10</td>
<td>20.4%</td>
</tr>
</tbody>
</table>

The results of this subsection outline the general characteristics of curation repositories: they are GitHub repositories that collect and organize alleged high-quality resources about technology industry, where articles and software projects are the most popular types of curated items; curated items come from both inside and outside of GitHub; and the richness of the description of curated items varies among different repositories.
The Owners of Curation Repositories

Repositories on GitHub are the results of collaborative efforts between owner and contributors. Owners create a repository, and contributors contribute to curation repositories through sending pull requests. For software repositories on GitHub, many are owned by an organization as a public space for their open source projects. GitHub organizations are group-owned accounts (Peterson, 2013). A repository that is owned by an organization is usually managed by a corporation or a group of developers (Peterson, 2013). For popular individual software repositories, owners are usually “coding rock stars”, i.e. GitHub users who have lots of followers and attract community attention, where GitHub users have interest in what projects they are working on (Dabbish et al., 2013). We were wondering if the same holds true for the owners of curation repositories.

Table 4-5. The number of GitHub users following curation repository owners prior to the creation of curation projects.

<table>
<thead>
<tr>
<th># Followers</th>
<th># Owners</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 10</td>
<td>22</td>
<td>46.8%</td>
</tr>
<tr>
<td>Between 10 and 100</td>
<td>15</td>
<td>31.9%</td>
</tr>
<tr>
<td>&gt;= 100</td>
<td>10</td>
<td>21.3%</td>
</tr>
</tbody>
</table>

For the 49 curation repositories, two (4.1%) were owned by an organization, and the other 47 were owned by individuals. For the 1,384 software repositories in our sample, 522 were owned by an organization (37.7%). It shows that the majority of the popular curation repositories were owned by individuals, which implies that most curation repositories could be successfully managed without group efforts.

The number of followers of an individual software developer has is an indicator of community status in GitHub (Dabbish et al., 2012; Marlow et al., 2013; Tsay et al., 2014). An example of a “coding rock star” is GitHub user “dhh” (Dabbish et al., 2012), who has thousands of followers. Prior work has found that the distributions of the number followers show a power-
law-like shape (Lima et al., 2014). However, the follower distribution of the owners of curation repositories have not been reported. In addition, Matthews et al. (2014) report that in a large enterprise environment, curators are usually the community leaders (Matthews et al., 2014). We are wondering if the same hold true in the social coding site as well. In the following analysis, we use the number of followers to indicate if the owner of a curation repository is a “coding rock star” or a community leader.

**Participation in Curation Repositories**

This subsection concerns about how GitHub collaborative features are used in curation repositories. For 40 out of 49 of the curation repositories, only the owners push changes to the main branch of the repository. In 48 of the curation repositories there are multiple contributors made contributions through pull requests. The number of contributors a curation repository has is presented in Table 4-6. The majority of curation repositories have more than 10 contributors.

Contributors of curation repositories appropriate the existing functions of GitHub to contribute to the repositories. In 29 curation repositories, there are contributors who made suggestions of resources in the issue tracker. Usually they fork the curation repository to their own space, make a modification, and send a pull request to the owner, which is the same process as comparing to the collaboration on software repositories on GitHub (Marlow et al., 2013).

Table 4-6. Number of contributors to curation repositories.

<table>
<thead>
<tr>
<th># of contributors</th>
<th># Curation</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 10</td>
<td>13</td>
<td>26.5%</td>
</tr>
<tr>
<td>Between 10 and 100</td>
<td>33</td>
<td>67.3%</td>
</tr>
<tr>
<td>&gt; 100</td>
<td>3</td>
<td>6.2%</td>
</tr>
</tbody>
</table>

For each curation repository, we examined all pull requests that were created before 8/31/2014 to extract the proportion of pull requests that have more than 2 participants involved.
At the time of this step of the analysis, one curation repository was no longer available due to copyright issues. One repository did not have any other contributors, and thus did not have any issues or pull requests. In addition, we eliminate 6 curation repositories that have less than 3 pull requests from this analysis in order Therefore, this analysis was performed on 42 repositories.

The results in Table 4-7 show that the contributions made to the majority of curation projects happen between only two people (contributor and owner). Rarely are there cases that a third person is involved in a pull request to evaluate the resource or to make a comment (Table 4-7). This might be effective for owners to manage the curation repositories since after contributors suggest an item, they can evaluate and then make a decision without introducing further steps. However, such mechanism might affect the quality of the alleged high-quality resources, since only two persons have examined the item.

Table 4-7. Percentage of Pull Requests that have more than 2 participants involved.

<table>
<thead>
<tr>
<th>Percentage of Pull Requests that have &gt; 2</th>
<th># repositories</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 10%</td>
<td>29</td>
<td>69%</td>
</tr>
<tr>
<td>Between 10% and 15%</td>
<td>7</td>
<td>16.7%</td>
</tr>
<tr>
<td>&gt; 15%</td>
<td>6</td>
<td>14.3%</td>
</tr>
</tbody>
</table>

To further understand how curation repositories are changed with pull requests, we retrieved all pull requests that were created before 08/31/2014 for the 42 curation repositories and the line changes for each using the GitHub API. We defined the following three types of pull requests, using “addition” to denote the number of lines added in a pull request and “deletion” to denote the number of lines deleted in a pull request:

- **Type Add**: in this pull request, addition > 0 and deletion = 0, or addition > 0 and deletion > 0 and (addition - deletion) > deletion
- **Type Delete**: in this pull request:
addition = 0 and deletion > 0, or
addition > 0 and deletion > 0 and (deletion - addition) > addition

- **Type Update**: the rest situation, where addition and deletion are similar.

![The repository counts vs. the percentage of Type Add, Delete, and Update of pull requests](chart.png)

Figure 4-2. The repository counts vs. the percentage of Type Add, Delete and Update of pull requests.

Following this definition of types of pull requests, for each curation repository, we calculated the percentage of each type of pull requests and plotted the results in Figure 4-2. The results show that in most repositories of this sample, Type Delete takes very small portion of the pull requests (less than 10%), while Type Add dominates most curation repositories, and Type Update takes some proportions. It indicates that curation repositories most receive contributions to add more resources, and sometimes there are contributions to change the existing ones, but rarely are there contributions to delete resources.
Discussion

Our results illustrate the characteristics of curation repositories regarding 1) their differences from software repositories on GitHub, 2) the topics and data provenance of curated items, 3) the leadership, and 4) the collaboration patterns. The emergence of curation repositories and their high popularity have significant implications, which are discussed in this section.

Internal Curation on GitHub

A relatively large proportion of curation repositories are internal-based, which suggests that one important function of curation on GitHub is indexing GitHub orientated resources.

GitHub has been reported to be a successful tool for self-hosting of software repositories and increasing effectiveness of collaboration (Dabbish et al., 2012; Marlow et al., 2013). As a result, many more developers and organizations began to host software projects on GitHub to allow contributions from others, while also contributing to other software projects. This action led the fast growing of the number of repositories on GitHub. The number of repositories on GitHub reached 10 million in 2013\(^\text{20}\). However, not all repositories hosted on GitHub are of high quality and can be appealing to software developers to use in their own projects or to contribute to.

Curation repositories provide valuable navigational support for software repository retrieval, with the help of GitHub features as well as human effort. And the high popularity of curation repositories and the large quantity of internally curated resources suggest that such attempt is highly welcomed in the software developers’ community. They are likely to save the time and efforts software developers spend in locating desired software repositories.

\(^{20}\) https://github.com/blog/1724-10-million-repositories
Meanwhile, as each curation repository usually supports a single (or several related) software development topic, it also raises the question about the scalability of curation practice. Particularly, with the fast progressing of software engineering, new programming languages, frameworks, and libraries are emerging daily, and thus the number of curation repository will grow as well. As a result, curation repositories as a whole will be fragmented. Some meta curation repository has already emerged\(^\text{21}\), which indexes and organizes curation repositories. However, the usage and effectiveness of such meta-lists are unknown. The user evaluation of such curation repository and design efforts for organizing curation repositories can be an interesting future research direction.

**Implications for the Owners of Curation Repositories**

In the review of the characteristics of the owners of curation repositories, we found that most of them are individuals rather than organizations. It implies that the creation and maintenance of a curation repository does not require group efforts. It also suggests that curation in social coding environment is different from the enterprise context, which tends to have a small leadership team that creates and maintains curation repositories (Matthews et al., 2014).

We were also curious if owners of curation repositories were leaders within the community and found that many were little known prior to their creation of the repositories, for they did not have many followers, which is an indication of leadership status in GitHub (Dabbish et al., 2012; Tsay et al., 2014). This is an interesting distinction considering that the GitHub community often favors the work of reputable, well-known developers (Dabbish et al., 2012). The reputation of these curation repositories shows that curators do not have to be community leaders within the social coding site for their curation repositories to be well received.

\(^{21}\) [https://github.com/sindresorhus/awesome](https://github.com/sindresorhus/awesome)
This result has important implications. Given the popularity and attention the curation repositories received, it is an opportunity for not well-known software developers to create good curation repositories and make an impact in the community. In addition, the curator may become an important role in software developers’ community, because 1) currently there is no easy way to deal with the information fragmentation, nor to address the difficulty in evaluating information (Storey et al., 2014; Storey et al., 2017), and 2) the software industry is changing fast, where new technologies are developed, and old ones are deprecated every day (Wu et al., 2015). It is likely that more curation efforts will be required in software developers’ community.

However, as shown in the results, as most owners of curation repositories are individuals, it also raises interesting questions on how well a curation repository can scale. The more popular a curation repository becomes, the more contributions it will receive, and the larger it becomes. It will become increasingly hard for the owners to add new curated items, track existing ones, and at the same time, to evaluate the ones suggested by contributors as the repository expands. It will be interesting to see if organizational efforts will be invested in a curation repository as it expands, or a community like open source projects, where there are core and peripheral members, will emerge around a certain curation repository.

**Collaborative Curation on GitHub**

The appropriation of GitHub for collaborative curation is of particular interest to this study because GitHub provides a number of collaborative feature, such as issue tracker and pull request mechanism, which becomes standard features in software practices.

Typical curation efforts include selecting, organizing, evaluating resources from multiple resources (Duh et al., 2012). In addition to these activities, the owner of a curation repository will also interact with other contributors to curate resources that match the description of the
repository. In general, curation repositories adopt the existing practices on GitHub intended for collaborative software development in which contributors send pull requests (or issues for some curation repositories) to the owner to submit a change to an existing file (specifically, to add a new resource hyperlink). The owner will then evaluate the resources recommended by contributors and decide whether to merge the change or not. In this way, curation repositories are collaboratively developed by a number of GitHub users. This kind of collaborative curation follows very similar contribution patterns as comparing to software repositories (Dabbish et al., 2012; Marlow et al., 2013; Tsay et al., 2014). Therefore, curation repositories not only adopted GitHub features, but also appropriate a part of the software practices on GitHub as well.

However, this type of appropriation of GitHub for curation differs from the enterprise context described by Matthew et al. (2014), which combines a number of tools to organize and curate resources to cope with information overload, and the community leaders usually curate the bulk part of the resources (Matthews et al., 2014).

Further, our results show that most collaborative curation happens only between two persons, the owner of a curation repository and the contributor. It raises some doubts on whether the opinions of two persons can be well representative for an artifact intended for a large community. It suggests that GitHub features are underutilized in terms of evaluating resources for reaching community consensus. In addition, most pull requests to curation repositories are adding new resources, lacking the deletions of the existing ones. It suggests that the contributions to curation repositories rarely consider whether existing ones are still up-to-date or are appropriated to be included in the list. In the long run, if a curated list keeps growing, it can increase navigational difficulties as well as affect the overall quality of the curated resources.
Design Implications

Our results demonstrate that curation repositories have become an important type of artifact developed in GitHub. The characteristics of such repositories have important implications.

From a design perspective, there are opportunities to design a better interface and provide a better user experience for curation repositories. Open source software project is a major type of resources for curation, and software projects are created, flourished, and perished all the time. Under current curation paradigm, there is no effective way to monitor if a curated item in a curation repository is under active development or not without manually checking. As suggested by recent literature, software developers leverage a set of features to make social inferences, for example, recent activity signals the activeness of a repository (Dabbish et al., 2012), and the number of stars indicates the community’s interest in a project (Tsay et al., 2014). As most curation repositories are a single page with lengthy content, and most of the time, the information contained in a curated item is brief, including only the name of the resources and simple description, these types of signals, such as the number of stars and activeness, can be appended to each curated item to help software developers evaluate curated items inside a curation repository.

Limitations

Our study is limited in the following aspects. Given the large volume of GitHub repositories, we were only able to examine a sample of them and elected to use the most popular ones as a logical boundary for our sample. Less popular curation repositories might have different properties. In addition, we applied content analysis and quantitative method to generate the characteristics of curation repositories. However, GitHub users’ motivations to create and
collaborate on this type of practice, and their perception of such repositories are also relevant and interesting, which should be the focus of future research efforts investigating this interesting phenomenon.

**Summary**

Curation on GitHub is an innovative appropriation of an existing tool in software developers’ community. In this paper, we studied the characteristics of curation in software developers’ community by investigating curation repositories in the following aspects: 1) the GitHub features used in curation repositories, 2) the characteristics of contents, formats, and owners of curation repositories, and 3) the collaboration patterns in curation repositories. Our results show that curation repositories make use of existing GitHub features to collect, organize, and retain resources about the technology industry. They centralize resources that are spread both inside and outside of GitHub. The comparison of activities between curation repositories and software projects illustrates that curation repositories have a more stable structure, receive more contributions from the community, and do not have multiple owners to lead the repository.

The emergence of curation on GitHub and its wide popularity has important implications. It suggests that curation may become an important way for software developers to communicate knowledge as the challenges of participating in multiple social media channels to face a large volume of resources are increasing (Wu et al., 2014; Storey et al., 2017). Also, curator role may become more important in software developers’ community and software developers can curate resources to make an impact. Last, there is potential for appending different pieces of information signals to each curation item inside a curation repository to reduce the navigational cost inside a curation repository.
Chapter 5
Curators Motivations and Experiences

This study addresses the following research question:

RQ2: What motivates software developers to curate resources? And what are curators’ experiences with curating on?

We conducted semi-structured interviews with 16 GitHub curators to better understand motivations to engage in this practice. In doing so, our study aims to investigate: (1) developers’ motivations that drive curation practices; (2) why GitHub is chosen for this purpose; (3) how curated resources are used; and (4) current limitations and potential future improvements for curation on GitHub. Our results suggest that curation practices on GitHub mostly grow out of software developers’ internal (altruism) and extrinsic motivations (personal needs and peer recognition). Software developers choose GitHub to perform curation practices mainly because this platform provides convenient tools and attracts vast groups of people with common interests. Software developers also benefit from curation in many aspects such as better software development support, efficient learning tools, and communication with the community. Further, curation represents a case that a collaborative working space is appropriated to an end-product for communicating high quality resources, suggesting GitHub repositories can be used for communication purposes to support the larger community of software developers. However, current curation practices are restricted by document format, curation process, and are bounded by GitHub features. The addition of built-in tools, such as navigation support within curation projects and automated resources for updates and evaluation, hold potential for improving current practices. Our study contributes to a better understanding of software developers’ motivation to curate resources and the nature of appropriating GitHub features for curation.
Methodology

To explore and understand software developers’ experiences appropriating GitHub for curation, we conducted a qualitative study with 16 curation project owners. In this section, we describe our recruitment procedure, interview protocol, and data analysis processes.

Participants Recruitment

To identify participants engaged in curation practices, we queried the GitHub search API on 12/07/2015 using the keyword “curated list” to search for curation repositories. The query returned 896 repositories hosted on GitHub. We recorded the owner’s user ID for each repository in the list, then we queried GitHub API again to fetch profiles with email addresses of each ID. The query returned 405 unique owners with email addresses, which we used to create a randomized list and sent 172 email invitations to curation project owners. Recipients were asked to engage in a semi-structured online text-based interview carried out via Facebook Messenger, Skype, or Google Hangouts. We began our recruitment process in early December 2015 and completed all interviews in late January 2016.

The resulting 16 participants included 15 males and one female with GitHub experiences ranging from six months to six years. Fourteen of the participants are professional software engineers, one was a graduate student, and one was a microbiologist. Eleven participants used the descriptive word “awesome” as the prefix to name their curation repository. The participants had a varying number of followers: five had less than 10 followers; eight had between 10 and 50 followers, and four had more than 50 followers. In the following section, we refer to individuals by participant number (from P1 to P16).
**Interview Protocol**

We conducted text-based online interview with participants, and each discussion lasted approximately 30 to 60 minutes. The interviews were semi-structured by the four general areas below.

- Motivations to curate resources,
- Reasons for technology choice (GitHub),
- How curated lists are useful,
- The limitations of current curation practices (on GitHub).

Questions were administered conversationally to engage the participants, and they were open-ended enough that we could pursue new topics raised by the participant.

Participants were interviewed in English. The interview scripts were then downloaded for analysis.

**Data Analysis Procedure**

We conducted our iterative analysis through four rounds of interviews allowing the first round of analysis to guide our second round of interviews and following similarly in the third and the fourth. Themes and codes were identified, discussed, and refined in this process (Lacey & Luff, 2001).

In the first round of analysis, we performed open coding on the responses (Strauss, 1987), grouping examples that are conceptually similar. For each subsequent round of interview, we compared concepts and categories that are similar to previous ones. And in this process, we continued to refine our coding scheme while also revealing new ones. We discussed the codes collaboratively and repeatedly. We concluded the study after reaching the point of theoretical
saturation, when categories, themes, and explanations repeated from the data (Marshall, 1996). Second researcher independently coded four sample interviews transcripts. Our analysis showed inter-coder agreement between the two researchers (kappa = 0.73).

In the process of coding, we recognized that some themes and categories are consistent with prior literature, i.e., curator motivations (altruism, personal needs, and peer recognition). Instead of developing new categories, we labeled them according to existing literature. The complete coding scheme is shown in Table 5-1.

Table 5-1. Coding scheme summary.

<table>
<thead>
<tr>
<th>Theme</th>
<th>Category</th>
<th>Count (n=16)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curator motivations</td>
<td>Altruism</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Personal needs</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Peer recognition</td>
<td>5</td>
</tr>
<tr>
<td>Reasons for appropriating GitHub</td>
<td>Familiarity with GitHub</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Relevant context and audience</td>
<td>13</td>
</tr>
<tr>
<td>Usefulness of curated list</td>
<td>Supporting work</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Learning a new topic</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Communication</td>
<td>10</td>
</tr>
<tr>
<td>Limitation of curation</td>
<td>Immature format</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Hard to maintain</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Difficult to market</td>
<td>5</td>
</tr>
</tbody>
</table>

**Results**

The results of our analysis describe curation practices on GitHub from the aforementioned four aspects, including (1) motivations to curate, (2) technology choice, (3) the use of curated resources, and (4) the current limitations of curation practices on GitHub. The analysis is presented through a count of themes present in coded interviews (Table 5-1) and representative quotes from each of the four themes.
Motivations for Curation

Internal factors (i.e., altruism, community identification, and intrinsic motivation) and external rewards (i.e., personal needs and peer recognition) are identified as motivating factors in software developers’ participation in open source software projects (Hars & Ou, 2001). In this study, our participants confirmed altruism (62.5%), personal needs (93.8%), and peer recognition (31.2%) motivated their participation in curation projects.

Internal factor – altruism

Participants reported that they engaged in curation practices because other community members might benefit from their effort. For example, P3 believed the high quality of curated resources could help beginners with programming:

“I see so many people when they take introductory classes in programming, they come to GitHub to get ready repositories...and that is overwhelming at first...so to get the started and motivated with programming I thought of collecting resources together in (P3’s curation project)” — P3

P6 wanted to help people who were in a similar position to himself:

“I’m a kind of remote engineer, then I want to create a list for someone tend to like me about product manager list, I just want to save some links for my learning purpose ... then public for someone if they’re in need”

— P6
External rewards

Personal needs and peer recognition form software developers’ external rewards derived from participating in open source projects (Ye & Kishida, 2003). These rewards also drive engagement in curation practices.

Personal needs were the most discussed reason for participation in curation (93.8%). Specifically, software developers reported that curation repositories improved productivity and enabled communication with others. Before creating curation projects, half of participants who were familiar with a particular set of resources relied on search engines whenever they tried to locate the URL of the resource. One important reason they chose to curate resources was to avoid such repetitive search efforts.

“Before making the repo I had to do research each time I needed a (P12’s curation topic). Now that I have a list, I just refer back to it when needed.”—P12

“I simply created my own list of the sites I found to be good. The idea really was to get out there scout for sites once and then be able to come back to a list without worrying about it having sites I found bad.”—P9

In addition, a curated repository has a permanent URL, which was reported as a convenient way to share resources with others who were outside of curation repositories. Participants stated that with curation projects, they only needed to point others to the URLs of their curation repositories. It was both convenient for them to share and for others to find. For example, P14 created the curated list so that she could conveniently share curated resources with others.
Peer Recognition surfaces as another important motivation for software developers’ participation in curation practices on GitHub.

The community of software developers on GitHub adopts a particular way to endorse curation projects. A highly reputable software developer on GitHub, Sindre Sorhus (9.2K+ followers), creates the “awesome” (repository name) project on 07/11/2014, which is a meta list of curated lists. It contains a community drafted “awesome manifesto”, which depicts guidelines and standards for curation practices, and requires that curation repositories conform to it if they want to be included in this meta list. The project currently has around 2,500 watchers, more than 35,000 stars, and approximately 4,000 forks, ranking the 2nd most starred repository created after 01/01/2014.

11 out of 16 participants used “awesome” as a prefix to their curation project name in an effort to conform to the naming convention as well as to indicate the quality of the content. 10 of them mentioned that they were inspired by the original “awesome” project. 4 of them hoped to get their curation repository indexed by it, and one participant’s curation project was already included in the “awesome” list, who felt a great honor (P10). P12 reported putting effort forward to improve his curated list to conform to the guidelines and standards as defined by the “awesome” project, stating “...with the Awesome endorsement I’m hoping it becomes a collection people trust” (P12). It demonstrated that our participants were putting efforts to align their goals with the larger community, i.e., conforming to the community standard for curating high-quality resources, and would like to be recognized by the community.

In addition, P14 reported that her involvement in curation efforts helped her obtain her current job, and P10 reported that a company approached him and wanted to collaborate with him.

---

22 https://github.com/sindresorhus/awesome
23 https://github.com/sindresorhus/awesome/issues/207
24 https://github.com/search?utf8=%E2%9C%93&q=created%3A%3E2014-01-01+stars%3A%3E1&type=Repositories&ref=earchresults
on his curated content. These rewards emerged as side-effects of curation efforts, not one of the guiding motivations for software developers to begin a curation project.

**The technology choice—why GitHub**

Compared to the GitHub platform, the features on sites like Wikipedia might be considered better suited for hosting curation projects by providing convenient editing and collaborating features. However, popularly used and referenced curation projects for software developers are predominantly hosted on GitHub, whose features and information structures are designed for source code hosting and project collaborating (Marlow et al., 2013; Storey et al., 2014) rather than creating and preserving lists of resources. In this section, we will address why curation has emerged as a common way to appropriate GitHub repositories.

**Familiarity with GitHub**

Participants reported that software developers’ existing knowledge about GitHub and its features, i.e., their strong media literacy (Storey et al., 2014) with GitHub, prompted them to choose this platform to host curation projects.

In general, software developers are familiar with GitHub’s text editing format (i.e., Markdown syntax) and are comfortable using it. P4 and P7 both claimed that “*GitHub was a tool that I was familiar with*” and “*so yes github would be a more natural tool to use.*” Specifically, software developers are accustomed to writing and formatting text contents with Markdown syntax. For example, P11 expressed that “*...I love write in markdown format!*”, and P5 considered that “*Github has a really easy way to write content in rich format (using Markdown) and view it.*”
“...as developer, I think github is the best place for developer to collaborate with other to build good resource.”—P15

Intimate knowledge about GitHub collaborating features is another factor:

“Github is a really good platform to collaborate. Anyone could come, fork it, extend it and ask me to ‘Merge’ it (update my list).”—P5

“...the advantage of using Github is other people can contribute easily.”—P4

Relevant Content and Potentially Interested Party

Participants also chose GitHub for curation because: (1) the curated contents were relevant within the GitHub context, (2) there was a large potential audience on GitHub, and (3) the GitHub community encouraged contributions.

A total of 15 out of 16 participants’ curation projects were related to software development practices. They considered GitHub just suitable as a platform for sharing software development related contents: “...(it is) the place to be for projects like this” (P2).

GitHub has attracted a large base of like-minded users when it comes to software development, which increases the chances of matching resources with an interested audience:

“GitHub has a very large audience/devs actively spending time in it, so it’s definitely the right place to publish a project such as this...”—P1

In addition, hosting curation projects on GitHub encourages contributions. GitHub has many collaborative features. It is a common practice on GitHub for users to contribute to other projects (Dabbish et al., 2012; Marlow et al., 2013; Wu et al., 2014). P12 reported that “GitHub
can target at the right audience and contributing is encouraged more...”. P8 claimed that “…enable other people to (freely) contribute to it is very important to me (and I think other curators also feel the same) so a Git hosting site is ideal.”

Participants also believed that other people on GitHub, who had more experiences and knowledge than themselves and other could enhance the repositories by contributing to lesser developed parts of the repository.

“The main reason is collaboration...I may have some resources but other people may have even better stuffs or ideas to share.”—P3

The use of curated resources

Curated resources are useful for software developers to support their work, learn about a new topic, as well as communicate with others.

Supporting work

Software developers rely on others’ work to accomplish their own projects. Participants reported that they used different curated lists, including their own, as bookmarks or references to quickly locate the resources they need.

“Before making the repo I had to do research each time I needed a (resources). Now that I have a list. I just refer back to it when needed. It serves as a good toolkit for future projects.” —P12
“I recently have created a Python repository and since I was not used with Python at all, I used awesome-python to know some libraries recommended by the community” — P11

In addition to supporting their work, participants also used the curation repository to keep track of high-quality resources in case they need them in the future. For example,

“If I used it or I’m planning to use it, I’ll add it there. If the resource is well written with tests and should be considered while selecting specific category, I’ll add it too... but also I add (resources) that I checked already and found it interesting for the future projects.” — P16

Learning a new topic

When first encountering of a certain topic, software developers often find themselves overwhelmed. The complex information scope in the software developers’ community makes it hard for developers to start tasks quickly. For example, P6 report that “when we start to learn new thing, there are many things, we cannot know what should to spend time on.”

A curated list that provides centralized peer-reviewed resources about a specific topic provides a starting point where developers know that they can find high-quality resources and begin learning the subject.

“...I’m an iOS engineer. But someday I like to learn Ruby, I just go to awesome-ruby and pick some recourse for beginner. Googling is not going to help us like that.” — P6
“So say if I starting to learn a new tool and need to get started quickly. I might go to the main awesome list and search for it.”—P5

**Communication**

Communication is essential among software developers in order to transfer knowledge between stakeholders, as well as facilitate learning, coordination, and collaboration (Storey et al., 2014). Curation serves important communication purposes, including reduction of communication costs and creating a shared knowledge space.

“I’m relative active in the meetup community in (P14’s location). Talking to people, there is always a lot of talk about what makes a good (P14’s curation topic). I created list so that I can point to other easily... I refer a lot of people to the list who are looking at improving their (P14’s curation topic).”—P14

One can share the content of resources with others in the current time as well, as reported by P7:

“...I sometimes encounter people who’ve watched (the topic of P7’s curation repository) and didn’t really like them, but my hunch is they haven’t seen the great ones, so I send them to check out my list to see if I can convince them otherwise...”—P7
Limitations of using GitHub for curation

In this section, we summarize constraints that our participants have encountered when carrying out curation on GitHub.

**Immature Structure and Format of Current Curated Lists**

The README.md file on GitHub typically includes an introduction to each project and current curation practices mainly rely on that single README.md file to list all curated resources. Sometimes a list may grow excessively long. Participants complained that “resources are not searchable (when on a list)” (P4), and it was cumbersome for them to navigate through a long list:

“The only thing sometimes that nags me is that some of them are very long, which in some sense defeats the purpose.”—P5

In a case where a curated list was too long, P6 created a shorter version of the same topic by selecting resources most important to him:

“there is another remote list...lot of stars, around 5k or more, but I find it that there are lots of resources, then when I look into, I’m scared of.

Then I want to create my own list, just something I think useful for most.”—

P6

Another issue raised is that the brief description of each item in the curated list (noted in Fig.1) can be incorrect, inaccurate, or misleading:

“Bad description doesn’t allow finding the required resource.”—P16
Further, although these curated lists are intended to be collaborative efforts (i.e., multiple people suggest adding, deleting, or updating entries), there is no intuitive way for an audience to express their opinions or raise uncertainties about resources, only modify content. One participant suggested including a rating system in the curated list to help audience filter resources:

“...maybe it would be better we could Like/Dislike the resources

...sometimes the resources are sorted by name when popularity would be a better option... something like this would give us an overview of how much important some entry in a list is for the community.”—P11

**Excessive Efforts to Filter and Maintain Resources**

It requires a lot of time and efforts to navigate in this complex information space where curation takes place and to filter a handful of good resources. P14 emphasized the time constraints for curation:

“Time. Time is hard...Digging through all of these resources takes time, and I’m usually pretty time constrained.”—P14

Due to the fast-changing nature of the software industry, old resources become outdated and new resources emerge instantly. Curation repositories require efforts to be simply maintained, including getting rid of the outdated resources, and adding state-of-the-art resources. P16 reported that one drawback of the current curation practices was that curated resources have “no quality update.”
**Difficulties for Marketing**

Although GitHub contains a vast and relevant user base, it does not provide mechanisms for a repository owner to distribute the list directly to the relevant audience. Our participants expressed that it was hard for them to target their repositories to users who were interested in the curation topic. For example, P10 conveyed his desire to recruit more contributors:

> “the only drawback is the lack of pull requests. I want more... (I want to) discover datasets I missed.” —P10

And P4 found that it was demanding to reach out to both potential collaborators and consumers:

> “While it's easy to host a project on github, you still need to put effort into marketing it, so you get other people contributing or finding it.” —P4

Unlike social media services such as Facebook, which automatically curates and recommends personalized content for each user, GitHub only contains technical features to allow users to search for information. If GitHub users are not aware of the existence of such curation projects, it would be difficult to find these resources in the first place. Therefore, admitting curation projects are embedded in the context of an abundant potential audience, they still lack mechanisms and features for marketing to parties of interest.

**Discussion**

Our study provides an in-depth view of curation practices on GitHub. We first assessed curator motivations in participating in curating activity, and compared them with motivations in open source participation literature. Then, we analyzed the reasons that GitHub was chosen for
curation purposes. Next, we evaluated the implications of curation repositories to software developers’ community. And finally, we uncovered the current limitations of curation practices. In this section, we generalize these main findings and discuss design suggestions with the hope to improve curation practice in the future.

**The Extrinsically Motivated Curators**

One primary motivation for engaging in curation is altruism, which is also widely recognized as an important motivation for software developers’ participation in open source projects (Hars & Ou, 2001; Lakhani & Wolf, 2003; Ye & Kishida, 2003). This finding indicates that helping behaviors may be a common reason why software developers are motivated to participate in some online activities. Thus, when designing systems for facilitating software development related practices, we should consider software developers’ desire to support each other.

Enjoyment-based intrinsic motivations are the primary reason that software developers take part in open source projects but were not specifically mentioned by the participants as a drive for engaging in curation. Researchers have learned that intrinsic motivations drive software developers to spend more time and effort on open source projects (Lakhani & Wolf, 2003), and it is positively reinforced by community recognition (Ye & Kishida, 2003). Therefore, many software developers commit themselves to open source projects for a relatively long time. At the same time, altruism alone is recognized as an unsustainable incentive for open source participation (Ye & Kishida, 2003). The comparison of motivations to curation with participation in open source projects leads to questions concerning curators’ long-term engagement, such as whether intrinsic motivations are involved in driving curators, whether altruism can sustain curator’s long-term participation in curation activities, and if not, whether there is a mechanism
that regularly feeds curators’ motivations to curate. The answers to these questions are beyond the scope of this study and requires further investigation.

**Leveraging GitHub as a tool for communication**

While GitHub is known as a tool for software projects hosting and collaborating (Dabbish et al., 2012; Marlow et al., 2013), and communicating knowledge in software artifacts (Storey et al., 2014). This study finds that GitHub is also a good tool for communication of socially generated resources for developers. Here, we describe the features that have supported this practice.

First, GitHub features allow curation repositories to be shared easily. With a robust version control system as we as uniquely assigned public URL for each GitHub repository, GitHub guarantees the integrity and durability of curated contents and enables easy sharing. These technical features make GitHub repositories ideal for communicating resources with others.

Second, GitHub attracts potential audiences who can contribute to curation repositories. By connecting to relevant audience group, GitHub allows others to suggest potential curated items and evaluate existing ones. As such, the emerging of curation repositories indicates that software developers’ community starts to utilize GitHub for communicating knowledge that is socially generated and maintained (Storey et al., 2014).

GitHub as a communication tool establishes its flexibility and reconfigurability. With a simple appropriation of its features, GitHub becomes a favorite tool intended for curation purposes in software developers’ community. Such reconfigurability will lead to other practices besides curation, which can further benefit software developers’ community. For example, GitHub users started to appropriate GitHub repositories to write and publish software
development related books\textsuperscript{25}, which accepted community suggestions as well as changes. Also, software developers initiated sharing training materials for others to discuss related matters as well as retrieving improvements\textsuperscript{26}.

\textbf{Curation to strengthen software developers’ community}

The crucial aspects to allow communities of practice to sustain and grow is to onboard new members and to educate the existing ones (Lave & Wenger, 1991; Wenger, 1998; Wenger & Snyder, 2000). The results of this study demonstrate that curation repositories on GitHub reflect these core utilities of communities of practices. Software development related resources are changing rapidly, and software developers usually rely on a number of services and channels, such as Stack Overflow and Twitter, to keep themselves up-to-date with the trend (Storey et al., 2014). By centralizing peer-reviewed resources in an active community of relevant audience, curation repositories create a reliable channel that simplifies the process of discovering high quality resources. They are likely to reduce the amount of efforts individual member of the community spent on locating and filtering the resources. In addition, as the curated resources are peer-reviewed, they are more likely to guide one’s learning of a certain topic than random resources encountered on the Internet. Thus, curation repositories optimize the way that resources are disseminated and consumed in software developers’ community, which in turn helps the community to grow.

\textsuperscript{25} https://github.com/getify/Functional-Light-JS/
\textsuperscript{26} https://github.com/kentcdodds/es6-workshop
Design Implications and future directions

Our analysis describes GitHub as a technical infrastructure that meets the needs of curation practices. In addition, an important lesson for curation technologies in the software developers’ community learned from this study is that the sociotechnical environment of GitHub plays a crucial part: millions of software developers have already been familiar with the features of GitHub so that media literacy won’t be a barrier for participation, and GitHub community is both the relevant audience and contains many potential contributors. Therefore, when designing technologies supporting curation in software developers’ community, the sociotechnical features of GitHub should be taken into consideration.

However, there is still substantial room for curation practices to improve. Our participants found that curating software related contents required a great deal of effort to filter resources and to actively maintain the existing ones. In addition, as the length of the curated list grows, it also creates navigational difficulties. The current conditions could be improved by (1) empowering curation with automated filtering tools, and (2) adding navigational support within a curated list.

Automated tools can reduce the amount of effort curators need to spend on curating processes. Curators currently do manual selection and evaluation of potential resources as well as eliminating outdated resources. Selections are usually achieved by employing search engines or following recommendations from others. Automated tools can help curators reduce the manual efforts spent on finding and maintaining resource lists. For example, some of our participants manually refer to third party tools to check resources status, such as last-updated-date. An automated tool that checks and filters resources according to query fields can largely diminish the noise and reduce time and efforts to select and evaluate resources. In addition, an automated tool
can also help maintain existing resources, by checking whether a software project is still under active development or it is deprecated.

Providing navigational support aims at solving the following issues: (1) lengthy curated list, (2) lacking a search function, and (3) lacking common themes across different curation projects. To be more specific, anchored table of contents, which is fixated on the screen, gives readers a clear structure of a document, as well as enables them to jump among sections. This change would make navigating a long list easier. Adding a search function within a curation repository, allowing users to query keywords of the curated items, can help users explore and find ideal resources promptly. Also, templates can provide common structure and themes in different curation repositories. For instance, our participants mentioned that one of the features they wanted for each curation repository to have was to include a beginner’s section, where they could easily find out hands-on resources. Curation repositories could adopt a template that includes commonly identified themes, so that users will be familiar with the structure of different curation repositories and thus locate resources more efficiently.

Future work should seek feedback from GitHub users who are consumers of resources in curation projects. Together with what we have learned from this study, we will design and implement tools to help curators select, evaluate, and maintain resource lists more effectively, and allow users to navigate and retrieve desirable resources readily.

Limitations

Our study was a qualitative investigation of self-reported curation practices and experiences of GitHub developers. More specifically, we did not carry out a controlled study to manipulate hypothesized causal relationships among constructs. Also, due to our limited sample size, cross tabulations among responses were unlikely to be generalizable to the larger population.
of curation repository owners. A general limitation of qualitative field methods is that some well-known approaches to validity, associated with positivist science, cannot be employed, such as construct validity, statistical validity, or predictive validity. For a qualitative research design such as ours, credibility and transferability are key validity issues: credibility is whether the results are believable, and transferability refers to the degree to which the results can be transferred to other settings (Guba, 1981; Hoepfl, 1997).

Credibility. We tried to ensure credibility by having two researchers code the data independently, and then calculating the kappa statistic to assess coding agreement. In addition, we used member checking in the interview process to have participants directly corroborate findings. These two approaches were encouraging and convergent, indicating very good credibility for our reported findings (Lincoln & Guba, 1985). We acknowledge that this is only a starting point in understanding curation practices in GitHub, but it succeeded in raising many issues that could be pursued now in more constrained research designs.

Transferability. There are several potential threats to the transferability of this study. First, the owners of the most popular curation repositories did not respond to our interview invitation. Given those celebrity curators’ massive audience base and extensively received contributions and attentions, their motivations and practices might be different from the general curators we focused in this study. Second, the way we recruited our participants was to use the keyword “curated list to search through repository descriptions on GitHub and identified repository owners as our potential interviewers. This search method is transparent and direct but may have missed curation repositories and owners that did not self-identify with our keywords. Follow-up research can expand the criteria for identifying repositories to further develop our findings. Finally, all of the curators we investigated were recruited on public GitHub, so our results may not generalize to closed source systems. This is another direction for subsequent research to develop our initial findings.
Conclusion

This study seeks to close a gap in the literature by providing a greater understanding of the motivations that software developers appropriate GitHub for curation, and their experiences with that practice. By conducting in-depth interviews with 16 participants about their curation experiences, we uncovered that curators were motivated by altruism, personal needs, and peer recognition, which were comparable to motivations to participate in open source projects. Whether these motivations support long-term participation in curation practice is yet to be discovered.

Curation repository is an appropriation of an online collaborative working tool, indicating that software developers’ community starts to leverage GitHub as a tool for communicating socially generated knowledge. It reflects the flexibility and reconfigurability of the tool. And other similar practices, such as sharing course curriculum on GitHub, start to surface.

In addition, curation repositories serve important functions of communities of practice. They support software developers’ work, guide learning through an engineering topic, and communications within the community. Curation practice strengthens software developers’ community and can help it grow.

Finally, current curation practices are limited by lacking standard formatting, tools for helping curators find and maintain existing curated resources, and reaching the target audience, which creates opportunities for future improvements.
Chapter 6

Understanding and Supporting User Experiences with Curation Repositories on GitHub

In the previous studies, we examined how GitHub features support curation in the software developers’ community and software developers’ motivations and experiences with curation practices. How curation repositories on GitHub are used by software developers was left uninvestigated. This study intends to close this gap by conducting a case study to examine software developers’ experiences with using curation repositories. The results have influenced the design of RepoHunter, the aim of which is to improve user experiences with curation repositories on GitHub by appending simple cues to curated items.

The inquiry is guided by the following research question.

**RQ3: What are software developers’ experiences with curation on GitHub from a user’s perspective? How can they be improved?**

The investigation starts with a survey of 38 software developers, revealing how curation repositories help software developers in sync with the community. In addition, popularity, activeness, and project status are the top three types of information cues that are helpful for navigation inside curation repositories and the evaluation of curated items. Afterwards, based on the insights gained with the survey study, it discusses the design and implementation of RepoHunter, which integrates the three types of information cues into the curation repository. A user evaluation from 32 participants of RepoHunter found its advantages in effective signaling and filtering resources and limitations regarding users’ preference for quality indicators. This chapter concludes with future design suggestions.
The information context includes the metadata, such as the who, what, where, when, why, and how, which allow community members to understand whether and how information is relevant to them. Such context provides rich details that make information items meaningful and memorable (Hoadley & Kilner, 2005).

A Case Study to Understand the Curation Repository

To gain a better understanding of curation repositories on GitHub, we conducted a qualitative study to collect users’ feedback to the awesome-go curation repository27, one of the most popular curation repositories when we sent out our survey in April 2016. The awesome-go repository was created in July 2014. It indexes resources about the Go programming language and has attracted over 12,000 developers to star the project. As we considered collecting detailed and unexpected feedback from respondents, it is appropriate to utilize open-ended questions to give respondents the freedom and space to give any answer, of any length and with any level of detail. In this way, respondents will not be confined to a limited set of answers that are available in a Likert-type survey design.

Participants and Procedure

We designed a web-based open-ended survey study using the Google Form survey platform. Our recruitment began in April 2016 and was completed the same month. To first generate a list of potential participants, we called the GitHub API and fetched profile information of software developers who had starred the awesome-go project. We then randomly sent out

27 https://github.com/avelino/awesome-go
survey invitations to 800 awesome-go followers. There were in total 38 participants who consented to partake in our user study.

Participants were asked to respond to open-ended questions about their feedback on the awesome-go project from various perspectives. The survey questions were guided by the following themes:

- What are software developers’ motivations to visit the awesome-go project?
- What types of resources do software developers find most useful?
- How does GitHub transparency support evaluation of resources in a curation repository?

Respondents were allowed to skip questions.

The average professional working experience for our participants is 6.2 years. Nine participants have a high school degree; 21 have a bachelor’s degree; seven have a master’s degree; and one has a doctor’s degree.

After the data collection was completed, we analyzed the responses to identify key findings. We performed the qualitative data analysis of answers with open coding schema (Strauss, 1987). Codes were discovered, discussed, and refined among the authors.

Data Analysis and Results

The response data were coded based on a set of rules developed from the questions asked, as well as information received from the responses. As the first step, we examined participants’ responses to the open-ended questions and identified significant concepts and aspects through content analysis of their answers. We then applied a more thorough process to code their answers for in-depth analysis. In this procedure, we examined the patterns of participants’ responses and attempted to correlate concepts generated in the coding process. By doing this, we expected to
better understand participants’ experiences with curation repositories on GitHub. More importantly, we expected these results to infer a better curation repository page design in practice.

Our analysis of software developers’ experiences with curation repositories on GitHub is unfolded in the following steps.

**Reasons for Visiting Curation Repositories**

The main reason why software developers visit curation repositories is to sync with the community, including following the trend, learning, and problem solving.

Seven mentioned that they wanted to keep themselves updated about the development topic, such as “I wanted to monitor what go projects are respected by the OSS community”. Sixteen participants reported that they intended to find learning materials, i.e., tutorials and best practices about the Go language, e.g., “I'm fond of Go, and one of the ways to improve myself was to learn from the existing go-projects, if not the best.” And ten participants tried to find resources and solutions to achieve specific development tasks; for instance, “Sometimes I need to look for a specific tool or anyway to get an idea of available projects in a specific domain without spending the whole day with a search engine.” In addition, 18 participants expressed general interests towards the topic, such as “[I have a] huge interest in go lang resources, and it is a nice list of curated resources.”

These reasons are comparable to software developers’ reasons to visit other social media services. For instance, software developers visit Stack Overflow to find learning resources and code examples (Nasehi et al., 2012). They visit Twitter to stay up-to-date with the latest technologies, tools, and changes (Singer et al., 2014). And they find open source projects to use and learn from on GitHub (Dabbish et al., 2012; Wu et al., 2014).
To be more specific, curation repositories take a different approach for software developers to keep up with the community: they centralize otherwise distributed resources. Besides, the resources included in curation repositories have been evaluated by other peers, filtering out low-quality resources. So instead of searching one site after another and filtering and evaluating resources themselves, software developers can visit curation repositories for peer-reviewed high-quality resources.

**Types of Information Discovery Behaviors in Curation repositories**

We examined whether software developers perform serendipitous discovery, fact discovery, rediscovery, and channel-based discovery with the awesome-go curation repository.

*Serendipitous Discovery and Fact Discovery*

Serendipitous discovery and fact discovery are comparable, but serendipitous discovery does not involve a well-defined goal, while fact discovery is performed with explicit goals (Voyloshnikova & Storey, 2014). We asked the participants the following two related questions in order to assess their serendipitous and fact discovery behavior:

1. *Did you have specific goals in mind for your last visit? What were they?*
2. *Did you unexpectedly find something useful to you?*

The first question is intended to identify if our participants’ most recent visit to awesome-go is a serendipitous discovery or a fact discovery behavior. The second question is designed to understand the effectiveness of these two types of information discovery behavior.

Thirty-five participants responded to both questions. The results are shown in Table 6-1.
Table 6-1. Serendipitous discovery vs fact discovery.

<table>
<thead>
<tr>
<th></th>
<th>Found something useful unexpected</th>
<th>Did not find anything useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serendipitous discovery</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>Fact discovery</td>
<td>18</td>
<td>5</td>
</tr>
</tbody>
</table>

It is evident from the table that fact discovery is the main type of information discovery behavior, and when a user of curation repository performs a fact discovery, s/he is more like to find something useful in the repository as well.

Rediscovery

Since currently there is no feature to support rediscovery of curated items, we are interested in whether our participants perform rediscovery on the curation repository itself, and how often they do so. We asked our participants the following question:

*How often do you visit the “awesome-go” project?*

Thirty-seven participants responded to this question. The results are shown in Table 6-2.

Table 6-2. Frequency of rediscovery for awesome-go curation repository.

<table>
<thead>
<tr>
<th>Frequency of rediscovery</th>
<th># of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed # of times</td>
<td>4</td>
</tr>
<tr>
<td>Less than once a month</td>
<td>6</td>
</tr>
<tr>
<td>Once a month</td>
<td>11</td>
</tr>
<tr>
<td>More than once a month</td>
<td>6</td>
</tr>
<tr>
<td>When needed</td>
<td>10</td>
</tr>
</tbody>
</table>
The results show that a very small portion of participants do not perform rediscovery on the awesome-go repository. It implies that software developers not only find the curation repository useful, but also develop some dependencies on it.

Channel-Based Discovery

GitHub provides a feature called “watch” for a software developer to subscribe to a repository to receive the updates of the repository. In order to determine whether our participants performed channel-based discovery, we retrieved all user names of the watchers of the awesome-go repository through GitHub API and asked our participants to provide us with their GitHub username. We examined if our participants “watch” the awesome-go repository.

Thirty-six participants provided us their GitHub username, and only one watched the repository. Therefore, the majority of our participants did not perform channel-based discovery.

Information Cues for Evaluating Curated Items

When asked what types of resources they found most useful during their visits of the awesome-go repository (33 responses), the top two categories of responses included software repositories (27 mentions) and text about the development topic, i.e., articles, books, and presentations (10 mentions). Notably, three participants mentioned that they were mostly interested in software repositories that are hosted on GitHub. Given text resources are usually located outside GitHub, where the information environment is not standardized, in this study, we only investigate information cues for software repositories on GitHub.

---

Specifically, we asked our participants to answer what signals, cues, and factors they would consider important and helpful when evaluating curated resources in awesome-go. Thirty-two participants responded to this question. The answers can be grouped into three categories (Table 6-3) – popularity, activeness, and project status. For instance, the number of stars signals popularity, the last pushed/commit date implies the activeness of a project, and the issue list indicates whether the project is used by others or has bugs currently.

Table 6-3. Information scent and associated cues software developers look for when evaluating curated items.

<table>
<thead>
<tr>
<th>Information Scent</th>
<th>Available cues</th>
<th># of mentions</th>
<th>Example quotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popularity</td>
<td>The number of stars/forks</td>
<td>25</td>
<td>“starts are quite good indicator that people at least are interested with the project (must be a reason for that).”</td>
</tr>
<tr>
<td>Activeness</td>
<td>Last pushed date, the number pull requests</td>
<td>14</td>
<td>“First of all I exclude dead projects whose last commit date was around at least 3-6 month before.”</td>
</tr>
<tr>
<td>Project status</td>
<td>Documentation, the issue list</td>
<td>5</td>
<td>“I tend to exclude projects with poor documentation.”</td>
</tr>
</tbody>
</table>

Although these information cues, i.e., the number of stars and last updated date, are available in each repository, as shown in Figure 1, they are not contained in curation repositories. Thus, when a software developer wants to check the status of many software repositories, s/he has to open each and switch back and forth. It becomes notably inconvenient when software developers try to compare several repositories.

In summary, curation repositories present an innovative way for software developers to keep themselves in sync with the community by centralizing high-quality resources distributed in different places. Software developers look for different types of information cues when evaluating curated items inside a curation repository. Due to the abundant resources inside each curation repository and the lack of some types of information cues, curation repositories create usability issues.
In the following sections, we will discuss our design and implementation of a prototype to address these issues and present a user evaluation of the prototype.

**RepoHunter: Bringing Information Cues to Curated Items to Support Curation**

The results suggest that the major drawback of the current design of curation repositories is the extra effort to find information cues for each curated item. To address this issue, we designed *RepoHunter*, which brings information cues to each curated item, to make them readily available.

## Continuous Integration

*Tools for help with continuous integration*

- *drone* - Drone is a Continuous Integration platform built on Docker, written in Go
- *goveralls* - Go integration for Coveralls.io continuous code coverage tracking system.
- *overalls* - Multi-Package go project coverage profile for tools like goveralls

## CSS Preprocessors

*Libraries for preprocessing CSS files*

- *c6* - High performance SASS compatible-implementation compiler written in Go
- *gcss* - Pure Go CSS Preprocessor.
- *go-libsass* - Go wrapper to the 100% Sass compatible libsass project.

## Data Structures

*Generic datastructures and algorithms in Go.*

- *binpacker* - Binary packer and unpacker helps user build custom binary stream.
- *bitset* - Go package implementing bitsets.
- *bloom* - Bloom filters implemented in Go.

Figure 6-1(a). The *original* version of the awesome-go repository.
Figure 6-2(b). The RepoHunter version of the awesome-go repository, including the number of stars and last pushed date for each curated item.

**Design Details**

The purpose of RepoHunter is to make it efficient and convenient for users to visit curation repositories. We intended to explore if drawing cues directly to curated items could be helpful for software developers. Specifically, we brought information cues of popularity and activeness to each GitHub repository in a curation repository. Figure 6-1 illustrates the original and the RepoHunter version of the awesome-go repository.

**Stars Count - Popularity Cue**

The “star” is a GitHub feature that allows software developers to bookmark a repository for future reference (Dabbish et al., 2012). The number of stars demonstrates how many software developers show interest in a repository, and thus it is a good indicator for popularity (McDonald
& Goggins 2013). We put the number of stars of a curated software repository right beside the resource name in *RepoHunter*.

**Last Pushed Date - Activeness Cue**

Our survey study above shows that the last pushed date is an important information cue for activeness, and the reason why is that software developers want to avoid projects that are no longer maintained. We put the last pushed date of a repository beside the number of stars (Figure 2).

The reason that we excluded project status, another type of information cue, as discussed above, is because the information is not readily available either from each repository or through GitHub API.

**RepoHunter Implementation**

*RepoHunter* works according to the following steps: (1) it retrieves the content of README.md of a curation repository and converts it into an HTML file; (2) it scans the HTML file, and for each curated item that is a GitHub repository, it queries the GitHub API to retrieve the current number of stars and last pushed date; (3) *RepoHunter* then attaches the info to the corresponding curated item, and (4) regenerates the README.md file with the modified HTML content. The generated file is also put on GitHub so that it contains nearly the same interface and environment as compared to the original curation repository, except for the additional information cues 30.

---

**Preliminary User Evaluation**

Following a similar procedure, we randomly sent survey invitations to 800 GitHub users who had starred the awesome-go repository, requesting them to use *RepoHunter* and to compare it with the original awesome-go interface. We were concerned with the question whether *RepoHunter* helps them achieve their goals for visiting awesome-go. Thirty-two users responded to our survey. The same data analysis procedures as in our first user study were applied.

**Results**

**General Attitudes toward RepoHunter**

The advantages of *RepoHunter* regarding efficiency and convenience have been confirmed by participants. Twenty-seven out of 32 participants highlighted the improvements of *RepoHunter*. For example, participants commented the following:

“*[RepoHunter] is better. It contains more details*”

“*[RepoHunter] gives me more information faster and let’s me judge the project’s long term strength. [RepoHunter], by a landslide.*”

“I prefer [RepoHunter]. That shows clearly the repo status we concern about”

Five out of 32 participants preferred the original page. Three of them thought the extra information was not a good indicator of quality, and two of them considered the number of stars and last pushed date noise which did not help them evaluate the curated items.
**Perceived Effectiveness of Information Cues**

Participants regarded the new *RepoHunter* design as superior in terms of effectively providing aggregated information in one page. Both the star count and the last pushed date were highlighted:

“*It would save me some time since I will not have to open several tabs to check the stars of each package, which is an important thing when I’m looking for a package.*”

“*Low stars/low activities are not interesting to me. Helps me filter quickly.*”

“*...[I] really like the most recently pushed.*”

**Future Improvement**

Participants also pointed out some direction to improve the format and layout of *RepoHunter* in the future. For example, participants suggested structuring the information into a table and putting the extra information in a separate column: “*It needs table structure now when there’s too much info for one line.*”
Discussion

Information Discovery in Curation Repositories

In this study, we analyzed the information discovery behavior of curation repositories using the framework developed by Voyloshnikova and Storey (2014). We found that software developers perform more fact discovery than serendipitous discovery in curation repositories, and fact discovery is likely to be more effective for them to find resources that are useful. Most software developers carry out rediscovery with curation repositories as well, which shows the usefulness and importance of curation repositories to them and also indicates that software developers start to depend on curation repositories. Channel-based discovery is not popular among users of curation repositories.

It is worth investigating the reasons why serendipitous discovery is less effective. Is it due to the overwhelming large amount of information included in a curation repository? Is it because the format of curated lists cannot help the users navigate inside the repository? Answers to these questions can lead to a better understanding of serendipitous discovery in curation repositories as well as yield insights to design better user interfaces for curation repositories.

Revisitation is one of the most common web browsing activities (Adar et al., 2008). Browsing previously visited web pages takes from 58% to 81% of the overall browsing activity according to prior studies (Tauscher & Greenberg, 1997; Cockburn & McKenzie, 2001; Voyloshnikova, 2015). Rediscovery is also found popular in visiting curation repositories. Many participants in this study accessed the awesome-go curation repository regularly. However, currently GitHub does not provide a feature to support rediscovery at the curated item level. Software developers cannot rediscover the curated items they visited before because currently there is no feature to record this information. Recording visited curated items and providing
interfaces for software developers to rediscover already browsed items can further assist rediscovery with curation in the software developers’ community.

As channel-based discovery is enabled in GitHub, it is surprising that nearly no GitHub user performs such information discovery behavior. One possible reason is that the watch function is designed around activity, i.e., if a user watched a repository, s/he will receive updates about the activities performed in a repository. However, curation repositories are content-focused, and activities might not be that meaningful for users of curation repositories. The exact reasons abide future research efforts.

**RepoHunter Implications**

The evaluation of RepoHunter yields promising results. Software developers find integrated information cues provide a better experience for visiting the curation repository, and it is helpful in filtering out unwanted items and in comparing similar options. In the meantime, our analysis also suggests further improvements.

First, options should be provided in curation repositories to allow hiding or displaying information cues, because while many software developers find popularity and activeness of a curated item useful, some also consider the extra information noise or not valuable. Allowing user preferences will aid in satisfying software developers’ different needs.

Further, some software developers were only interested in curated items that meet some certain criteria, such as the most popular ones or active ones. A query function should be provided in curation repositories so that software developers can choose to look at only the ones that match their query criteria.

In summary, this study seeks to add to the research literature by providing a greater understanding of curation repositories on GitHub. The development of design principles as well
as the preliminary evaluation of our proposed RepoHunter could improve curation practices on GitHub. In the future, we intend to apply our proposed RepoHunter framework to other popular curation repositories on GitHub and other platforms to provide the research community with comparable results across a spectrum of choices. We also would like to explore opportunities for future collaboration with GitHub and software developers to conduct large-scale field experiments in the context of naturally occurring user practices.
Chapter 7

Discussion

In the previous three chapters, we applied mixed methods to make an inquiry into the curation behavior in the software developers’ community. Our investigation yields insights in understanding how social coding features on GitHub are leveraged in curation repositories, the curators’ motivations and experiences with curation on GitHub, and what the users’ perception of curation repositories are as well as how to improve a curation repository to provide better user experiences. The main findings include the following.

The main purpose of curation repositories is to select and organize high-quality resources located inside and outside GitHub. Articles about the technology industry and other open source projects are the most common type of resources included in a curation repository. Software developers collaborate with each other in curation repositories in a way that is different from how they do so in software repositories.

Curators, i.e., the owners of a curation repository, in the software developers’ community are motivated by altruism, personal needs, and peer recognition. They choose GitHub for curation purposes mostly because of media literacy, i.e., their familiarity with GitHub, and relevant content and audience. Curators use curated lists to support work, to learn new topics, and to communicate knowledge in the community. Curators also report some limitations with curation on GitHub. The formatting of curated lists is immature: there is no standard way to make a curated list and some features, like allowing the audience to quickly express their opinions and attitude towards a curated item, are lacking. Adding and maintaining resources inside a curation
repository require significant manual efforts. Although GitHub provides a relevant context and potential audience, it is still hard to make their curation efforts known to the community.

Software developers visit curation repositories to stay up-to-date with a topic, learn new skills, and find help for problem solving. They perform different types of information discovery behavior, including serendipitous discovery, fact discovery, and rediscovery. Information cues, including popularity, activeness, and project status, are found useful for software developers to evaluate a curated item. We designed RepoHunter, which appends popularity cues and activeness cues to curated items, and did a preliminary user evaluation. Most of our participants preferred RepoHunter and considered it more effective for them to browse within a curation repository and to find ideal curated items.

The emergence of curation on GitHub and the results of our studies have significant implications for the software developers’ community, and it has a lot of potential. Future research directions which will be discussed in this section.

Supporting the Software Developers’ Community with Curation

Our results and analysis show that most curation repositories on GitHub select, organize, and preserve different types of high-quality resources, grouping into different categories, that are useful for software developers. The wide popularity of curation repositories indicates that they are well received in the software developers’ community and attract enormous attention. It is likely that curation repositories will become an important way to support the software developers’ community.

Hoadley (2012) pointed out four techniques by which technology can support communities of practice: (1) linking people with others who have similar practices, (2) providing a shared repository of information resources, (3) providing tools for discussing with others, and
(4) providing awareness in a community of the information context of various resources. Curation on GitHub supports software developers’ communities in all the points as brought up by Hoadley (2012).

For the first point, GitHub, being an online service for source code hosting and collaboration, provides not only the “following” feature for software developers to subscribe to the activity stream of people of interest, but also a number of other tools, such as an issue tracker, a version control system, and pull request interfaces. Curation repositories on GitHub thus inherit these features to provide powerful features for software developers to connect with like-minded users and enable them to engage with conversations.

For the second point, curation repositories are information resources by nature, where software developers can find worthy resources with serendipitous discovery or perform fact discovery to identify an exact resource that solves a specific engineering problem or to learn a particular technique.

For the third point, our studies find that curation repositories use both an issue tracker and the Pull Request function, where both allow discussions by a group of people.

For the last point, our results also show that many curation repositories include author information and description for curated items. In addition, information cues, such as the number of stars, forks, and most recent update time, are used intensively by software developers when evaluating a curated item. These belong to the information context as well. Moreover, the aim of RepoHunter is to simplify how software developers can retrieve the relevant information context for each curated repository.

In addition to the points mentioned by Hoadley about the perspective that technologies support communities of practice, our analysis also shows that one defining feature of curations on GitHub is that it enables the collaboration among community members to create the information repository collectively. As shown in this study, Pull Request is commonly used in many popular
curation repositories. As a result, many curation repositories are the collaborative efforts of a group of software engineers.

Furthermore, as we discussed in Chapter 2, software developers are increasingly active in participating in a set of different social media sites (Singer et al., 2013; Storey et al., 2014; Storey et al., 2017), which creates vast opportunities for software developers to find information that is relevant and useful, but also introduces many burdens and challenges, such as (1) the fragmented resources spread over a set of social media sites; (2) the overhead to learn and master different social media channels; and (3) the difficulties in evaluating the quality of information in a large information space (Cha et al., 2007; Storey et al., 2017).

Curation repositories on GitHub are likely to be a starting point to address these challenges. Curation repositories centralize fragmented resources from all over the Internet. They are located in GitHub, a site many developers are already familiar with, where no other media literacy is required to master the tool. They have involved a collaborative human effort to evaluate the quality of the curated content. In this way, curation repositories become an important way for the software developers’ community to communicate quality resources so that millions of developers do not have to follow different social media sites and filter resources themselves.

In summary, curation in GitHub supports the software developers’ community by linking people with similar practices and provides shared information repositories, tools for discussion, and an information context. In addition, our results also suggest curation repositories on GitHub support the community by allowing communicating socially constructed knowledge. It is a starting point to address the challenges brought by the social media era.
The Curator Role

The curator is an emerging role in the software developers’ community. They create curation repositories on GitHub with a list of high-quality resources that they find useful for a specific topic. They maintain the repository by constantly updating the list with the new resources they find, or with the ones suggested by other software developers in the community.

The curator role is comparable to the triager role as reported in recent literature. Triagers usually apply manual efforts to filter and augment bug/issue reports in a software project, and they are effective in reducing developers’ loads by eliminating irrelevant reports and filling in missing information (Xie et al., 2013). In the Mozilla community, triagers are dedicated software developers who focus only on filtering and sorting bug reports (Jing et al., 2015).

Both triagers and curators sort and organize information provided by a group of people to another group. Triagers filter and improve the quality of user-reported bugs and issues. Curators organize and maintain high-quality resources that are produced by other software developers. In this sense, they assume broker-like roles, but they control different directions of information flow. Bug and issue reports are usually generated by peripheral participants of an open source community (Crowston & Howison, 2005). Triagers filter information generated by the users for the core developers of a community. Thus, triagers help the information flow from the outer community to the inner community. On the other side, curators usually select and index high-quality resources, relating resource producers with resource users. Hence, curators aid the information flow from developers to users, which is the opposite of what triagers do. The emerging of such roles shows their importance of building connections within a community as well as disseminate knowledge among community members. Similar roles are likely to appear in the future as more information is generated and the participating environment is becoming increasingly complicated.
The popularity of curators that serve as broker and help the information flow rises interesting questions concerning their professional trajectory. Brokers are part of different groups and control information flow in a community. They often bargain for better terms because of their unique positions (Burt, 2004; Burt, 2005; Van Liere, 2010). However, the existing studies of brokers either happen in the cooperate environment (Burt, 2004; Burt, 2005) or community networks (Carroll, 2012), and the brokerage results from establishing social connections, which leads to social capital gain (Carroll, 2012; Burt, 2005; Van Liere, 2010). In contrast, curators broker information by creating artifacts in the software developers’ community. However, the questions whether curation helps curators bargain for better terms in general, whether curators establish social ties, and how long such social ties exist as a result of curation require future investigation.

The Social Coding Service for Curation

Media literacy, the ability to access, analyze, evaluate, and create messages in a variety of forms (Livingstone, 2004), is an important challenge that software developers are facing in the social media era (Storey et al., 2014). Software developers need to learn the conventions as well as technical nuances of new media services that appear every once in a while, in order to use them effectively (Storey et al., 2014). The appropriation of GitHub for curation largely relieves the burden for software developers to acquire new media literacy. GitHub was launched in April 200831 and became widely popular in 2011. It gradually became the standard for software developers to host source code and collaborations. As a result, a significant part of the members in the software developers’ community is already familiar with GitHub features. As shown in our second study, one of the important reasons for choosing GitHub for curation purposes is because

31 [https://en.wikipedia.org/wiki/GitHub](https://en.wikipedia.org/wiki/GitHub)
software developers are familiar with the services, such as creating and maintaining repositories with Git and generating content with the markdown syntax. Therefore, appropriation of the social coding service, i.e., GitHub, is an effective application of the media literacy of the community members to support the community itself.

In addition to the media literacy, many concrete features of the social coding service support curation in specific ways.

In curation repositories, the name and description of a repository is appropriated to show the quality of the curated resources. The main public page of a repository is appropriated to organize and present the content of curated resources. The version control system, i.e., Git, and issue tracker are appropriated to maintain the content of the curated resources, as well as collaborate with other community members. In our last study, we also show that appending cues that are available in GitHub, such as the number of stars and the last active date, to curated items can provide better navigational experiences within a curation repository.

Our results and analysis suggest that many features of the social coding site are desirable for curation tools as well. In addition, understanding the existing media literacy of community members and design tools that leverage them is likely to have a positive influence on the tool adoption.

**Future Directions**

**Supporting Curation Community**

*Technology stewards adopt a community’s perspective to help a community choose, configure, and use technologies to best suit its needs*

– Wenger (2009)
As shown repeatedly in this thesis, curators on GitHub are technology stewards in the software developers’ community who appropriate the social coding service, i.e., GitHub, for curation to support the community itself.

At the same time, it is an emerging practice that only surfaces in the software developers’ community and has become popular in recent years. It includes non-subtle work, such as selecting a list of resources, organizing different resources into meaningful categories, maintaining the list to keep the content up-to-date, and collaborating with other community members. As shown in the second study, curators encounter difficulties such as struggling to find the best format to present the curated content, spending excessive efforts to manually filter resources, and having frustration to market their curated repositories to people in need.

A community of practice that gathers curators in the software developers’ community to share, learn, and exchange the knowledge about curation itself can further strengthen the practice and the role in the software developers’ community. Research efforts in investigating the forming of the community of practice for curators, the technologies that support such practice, and processes that onboard new members will be beneficial in helping the development of the practice and the community in the long run.

**Software Developers’ Information Needs in Social Media**

Existing literature about software developers’ information needs usually concern about what information software developers seek while they are engaging with a specific software practice, such as software maintenance tasks (Ko et al., 2006), daily information needs during work time with collocated colleagues (Ko et al., 2007), or reporting a bug (Breu et al., 2010). However, as reported by recent literature, besides engaging in software practice, software developers also spend significant amounts of time in social media channels, such as Twitter,
GitHub, and Stack Overflow, to stay updated with the latest news, to make connections with like-minded, and to find answers about specific questions.

Particularly, as we showed in the third study, one prominent purpose for which software developers visit curation repositories is learning. They want to learn the landscape of a technology, a specific piece of library or framework, or solutions to a particular engineering problem. Learning is identified as a major need for software developers’ participation in social media (Singer et al., 2014). However, there is rarely literature discussing software developers’ information needs during the learning activities in social media. Questions such as what software developers like to learn in daily social media participation, what kind of resources are most suitable to satisfy their learning needs, and how to organize and present different existing resources to software developers to achieve the best learning outcome are not answered yet, and how curation can help achieve these purposes is also not investigated, so this presents great potential for future directions.
Chapter 8

Conclusion

Software developers who actively incorporate social media sites in their daily practices, such as learning and working, are facing a set of challenges, including keeping up with the latest technologies, navigating and participating in a wide range of social media channels, and consuming a large quantity of information (Storey, 2017). Curation, which is the activity to select, evaluate, organize, and maintain a set of resources for preservation and future use, has become popular in the software developers’ community in recent years. In this thesis, we present three studies on curation to examine how features of GitHub support curation, why curators in the software developers’ community started this effort in the first place, and how users perceive curation repositories. Our results and analysis show that curation repositories in the software developers’ community alleviate such kinds of challenges by maintaining and sharing a repository of high-quality resources for the other community members to navigate and use.

Software developers are not the only group facing the issues of quantity and quality of information in this social media era. Both Twitter and Pinterest contain large quantities of different communities of people, let alone Facebook, the largest social networking site on Earth. These communities are encountering a number of challenges, with “fake news” being a prominent one.

Content on social media sites can be published among users without fact checking and editorial judgment, and individual users can reach as much audience as large media agencies (Allcott & Gentzkow, 2017). Especially during the 2016 election, much fake news was circulated on social media, and many people who saw the fake news believed it (Allcott & Gentzkow,
2017). If such community provides tools to a group of curators who are dedicated to fact checking of the news and to inform the audience, the fake news might not be prevailing.

In summary, as our lives are intertwined with social media, many challenges surface, such that everyone has to navigate through a large volume of information, the quality of which is not guaranteed. Curation with the support of social features and collaborative editing allow a group of people to select, evaluate, organize, and maintain a repository of high-quality resources, which is beneficial for the larger communities. We believe curation activities and technologies that support curation will also gain momentum among many online communities besides the software developers’ community.
References


In Proceedings of the 2nd international workshop on Web 2.0 for software engineering (pp. 31-36). ACM.


Duggan, M., & Smith, A. (2013). 6% of online adults are reddit users. Pew Internet & American Life Project, 3, 1-10


Handel, M., & Herbsleb, J. D. (2002). What is chat doing in the workplace?. In Proceedings of the 2002 ACM conference on Computer supported cooperative work (pp. 1-10). ACM.


McDonald, N. and S. Goggins. 2013. Performance and participation in open source software on Github. In *CHI '13 Extended Abstract* (pp. 139-144).


Pham, R., Singer, L., Liskin, O., Figueira Filho, F., & Schneider, K. (2013). Creating a shared understanding of testing culture on a social coding site. In *35th International Conference on Software Engineering* (pp. 112-121). IEEE.


VITA

Yu Wu
1144 Lily Ave, Sunnyvale, CA 94086
(814)-321-8276
yuw132@psu.edu

Education
The Pennsylvania State University 08/2018 (Expected)
Ph.D. in Information Sciences and Technology
Tongji University 04/2010
M.E in Computer Science and Engineering
Tongji University 06/2007
B.E in Computer Science and Engineering

Research Experiences
The Pennsylvania State University 05/2012 – 05/2015
Graduate Research Assistant
Avaya Labs 05/2014 – 08/2014
Research Intern
Tongji University 09/2007 – 03/2010
Graduate Research Assistant

Recent Publications