Evolving Software Ecosystems: The Role of Community Dynamics in Shaping Software Extensions

Elmira Onagh

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

YORK UNIVERSITY
TORONTO, ONTARIO

June 2025

Abstract

As software ecosystems (SECOs) grow across domains, understanding how tools evolve and differentiate functionally is critical for innovation. This manuscript-based thesis explores the evolution of the software ecosystem and its influence on developers' motivations to extend their software products in two ecosystems.

In the first part, we focus on the evolution of open-source software by analyzing 6,983 GitHub Actions on GitHub Marketplace, revealing a widespread functional redundancy. A graph-based analysis of version histories and release patterns identifies early contributors and offers strategies to reduce duplication and align tools with emerging trends.

In the second part, in collaboration with industry partners, we examined proprietary software products, focusing on functional maturity, in particular AI-related features in 116 patient-centric healthcare applications. We find that 86.21% of apps remain in early AI adoption stages, indicating limited advancement toward AI integration.

Together, these studies introduce a generalizable, data-driven framework for analyzing functional evolution across domains.

Declaration of Authorship

I, Elmira Onagh, declare that this thesis titled, "Evolving Software Ecosystems: The Role of Community Dynamics in Shaping Software Extensions" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed: Elmira Onagh

Date: June 9, 2025

Acknowledgements

The completion of this thesis would not have been possible without the support, guidance, and encouragement of many individuals, to whom I am deeply grateful.

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Maleknaz Nayebi, for her steadfast support and invaluable guidance throughout my graduate studies. Her expertise, patience, and mentorship have greatly influenced both my academic development and personal growth.

I am profoundly thankful to my beloved family and friends for their unconditional love, encouragement, and support during this journey. Their understanding and constant belief in me have been a continuous source of motivation.

I would also like to thank the department for providing the resources and facilities essential for conducting my research. Their support has been instrumental in helping me overcome the challenges of graduate school.

Finally, I extend my heartfelt appreciation to my lab mates for their collaboration, camaraderie, and inspiring dedication to research. Working alongside them has been both enriching and enlightening, and I am grateful for the knowledge and experiences we have shared.

Table of Contents

List of Figures

Abbreviations & Acronyms

AI	**A**rtificial **I**ntelligence
AIMMs	**A**I **M**aturity **M**odels
API	**A**pplication **P**rogramming **I**nterface
CI	**C**ontinuous **I**ntegration
CD	**C**ontinuous **D**eployment
EMR	**E**lectronic **M**edical **R**ecords
FM	**F**oundational **M**odels
HFM	**H**ealth **F**oundational **M**odels
LLM	**L**arge **L**anguage **M**odel
ML	**M**achine **L**earning
NLI	**N**atural **L**anguage **I**nference
NLP	**N**atural **L**anguage **P**rocessing
OI	**O**pen **I**nnovation
OSS	**O**pen **S**ource **S**oftware
RAG	**R**etreival **A**ugmented **G**eneration
RE	**R**equirements **E**ngineering
SDLC	**S**oftware **D**evelopment **L**ife **C**ycle
SE	**S**oftware **E**ngineering
SECO	**S**oftware **ECO**system
SoS	**S**ystems-**o**f-**S**ystems

Chapter 1

Chapter One: Introduction

Software ecosystems (SECOs) have become foundational to how modern software is developed, distributed, and maintained [1, 2]. These ecosystems are often built around multi-sided platforms that enable collaboration between tool developers, users, and service providers, facilitating innovation through component reuse and modular growth [3, 4]. Prominent examples include open-source platforms like GitHub and proprietary mobile app stores such as Google Play and Apple's App Store [5, 6, 7]. Although prior studies have explored the role of SECOs in driving innovation and influencing business decisions [1, 2, 3, 8, 9, 10, 11, 12, 4], critical questions remain regarding how functionalities emerge, mature, evolve, and compete within ecosystems.

1.1 Empirical Contexts

The GitHub Marketplace represents an evolving, developer-driven SECO that supports the distribution and reuse of automation tools, particularly GitHub Actions [5, 13]. These tools are defined through `YAML` workflows and are commonly employed for Continuous Integration (CI), Continuous Deployment (CD), code analysis, testing, and infrastructure management.

The marketplace has grown rapidly, from 7,878 Actions in April 2021 to over 22,000 in March 2024, indicating significant tool proliferation [13]. However, this growth is accompanied by high functional redundancy, particularly in categories like CI, where multiple tools offer near-identical capabilities. Developers often choose between reusing

existing tools, customizing forks, or developing new yet functionally similar tools [14, 13]. This behavior suggests complex innovation dynamics, influenced by both technical and social factors.

Furthermore, the open-source nature of GitHub Actions hosted on GitHub Marketplace enables transparency: researchers can access source code and metadata on development activity, composition patterns, and community interactions [15, 16]. This makes GitHub Marketplace a rich case for studying SECO evolution, redundancy, tool composition, and competitive differentiation strategies.

In contrast, mobile app stores such as the Google Play Store serve a user-centric SECO, where applications are consumed primarily by end users with limited technical knowledge [17, 7, 18]. Understanding the app store ecosystem is essential when developing mobile applications. Mining app store data helps identify widely adopted and emerging features, supporting data-driven decisions in software evolution, feature maturity and reuse [19, 20]. Comparative analysis of competing apps informs market entry strategies, while the multi-sided nature of app stores enables collaboration among developers, users, and service providers to reduce costs and enhance app value [21, 7].

Simultaneously, Artificial Intelligence (AI) has emerged as a transformative force across industries. Foundational Models (FMs), which are the large-scale AI models trained on diverse and extensive datasets, have enabled general-purpose capabilities with minimal domain-specific tuning [22]. These models can be tuned to a variety of downstream tasks. For instance, in healthcare, FMs are applied to clinical note generation, diagnostics, and personalized health interventions, promising a new generation of patient-centric e-Health solutions [23, 24].

External disruptive innovations are increasingly affected by markets such as app stores, often more rapidly and significantly than others [25, 26, 27]. In particular, the digital health domain has seen growing interest in integrating AI capabilities, including FMs, into digital health (e-Health) applications. These applications aim to provide personalized recommendations, symptom checkers, medication reminders, and conversational agents to enhance patient engagement. However, integrating AI capabilities

faces many challenges, including data privacy concerns, lack of explainability, regulatory constraints, and potential algorithmic bias [24].

App store metadata offers a valuable lens for understanding how functional features emerge and mature. By mining app descriptions, user reviews, and update logs, researchers can infer which features are widely adopted, which are emerging, how these features are maturing and how specific features such as AI capabilities are being positioned in terms of value and functionality [28, 19]. This mirrors the concept of feature reuse and ecosystem-aware development strategies in SECO literature [29, 20].

This work builds upon the existing body of literature developed over the years [30, 31, 5, 13, 32, 6, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 7, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 41, 46, 63, 64].

1.2 Challenges and Research Gaps

Despite operating in different contexts and serving different user groups, both the GitHub Marketplace and Google Play Store reveal similar structural and evolutionary challenges:

Redundancy and Innovation Trade-offs: In GitHub Marketplace,overlapping functionalities among Actions are common [13]. Similarly, in e-Health, multiple apps replicate similar features without building on shared foundations. These trends raise questions about the incentives and mechanisms behind functional duplication versus reuse [52, 44, 37, 65, 55].

Transparency and Data Access: GitHub provides source code and integration metadata, allowing for fine-grained analysis. In contrast, app store platforms are often black boxes, obscuring how AI models are used and evaluated [18].

Functional Maturity and Evolution: Both platforms enable studying the evolution and maturity of functionality across time. In GitHub, we can observe feature evolution via workflow and version history analysis; in e-Health, we assess functional shifts through app version histories and descriptions.

User and Provider Asymmetry: GitHub's users are technically proficient, which fosters open collaboration and rapid innovation [66, 13]. App store users, especially in healthcare, lack this expertise, increasing reliance on black-box AI and requiring higher standards for usability and trust [67, 43].

AI Maturity Assessment: There is a lack of standardized metrics to gauge the depth and reliability of AI integration in SECO. Existing maturity models (e.g., Gartner's) provide a starting point, but empirical studies applying them to real app data are sparse [68, 69, 68, 70].

1.3 Research Objectives

This thesis aims to investigate the emergence, maturity and evolution of software functionalities within SECOs, focusing on both open developer-centric platforms and user-centric mobile ecosystems. Specifically, the objectives are:

Objective 1: Investigate the emergence, evolution, and differentiation of functionalities within open-source software ecosystems (SECOs) by analyzing the historical development of GitHub Actions. This includes identifying patterns of functionality reuse, expansion, and the influence of early tool providers in shaping the ecosystem. We addressed this objective in Chapter 2.

Objective 2: Examine the functional maturity, particularly AI-related capabilities, within black-boxed SECOs by conducting a case study of mobile health applications, uncovering how functionalities are integrated and evolve in closed systems. This objective was addressed in Chapter 3.

Objective 3: Deliver actionable insights into the maturation and competitive dynamics of functionalities across digital ecosystems. These insights will support data-driven decisions for tool development, functionality reuse, and strategic differentiation tailored to each SECO type. We addressed this objective across Chapters 2 and 3.

1.4 Thesis Overview

This thesis is structured into four chapters, each contributing to a broader understanding of how functionalities evolve, overlap, and innovate within software ecosystems in alignment with the objectives outlined in Section 1.3. The work is divided into two empirical case studies and a conclusion chapter.

1.4.1 Chapter 2: Extension Decisions in Open Source Software Ecosystem

Chapter 2 investigates the GitHub Marketplace as a dynamic and open software ecosystem where developers build and share automation tools, particularly GitHub Actions. The study explores how Action functionalities and features evolve within this ecosystem by analyzing a large dataset of automation-focused Actions over time to address **Objective 1** and **Objective 3**. Specifically, we address:

- The functional overlap among Actions via graph-based modeling techniques.
- The identification and evolution of functional relationships, focusing on changes between two time snapshots.
- The role of tool providers, particularly "early contributors," in driving innovation and reuse.
- The mapping of these observed patterns onto established SECO migration models from prior literature [28].

Methodologically, this study employs `YAML` file parsing, Natural Language Processing (NLP), and network analysis using `NetworkX` [71] to construct and analyze a functional network of GitHub Actions. We compiled a dataset of 6,983 Actions from the GitHub Marketplace, capturing two temporal snapshots to enable longitudinal analysis. For each Action, software features were extracted using a fine-tuned Large Language Model (LLM), allowing for consistent and scalable identification of functional

attributes across the dataset. Based on these extracted features, we formalized functional relationships among Actions and modeled them as a network to detect patterns of overlap, similarity, and compositional structure.

To examine ecosystem dynamics, we studied the evolution of these functional relationships over time, tracing how tools converged or diverged in capability. Furthermore, we identified 3,869 distinct tool providers and analyzed the role of early contributors, those whose initial tools became functionally connected to others in later stages, in shaping the trajectory of the ecosystem. By uncovering patterns of functional redundancy and mapping the evolution of offerings, this chapter provides valuable insights to inform strategic decisions around market entry, tool reuse, and differentiation. The findings offer developers and platform stakeholders a roadmap for aligning future offerings with emerging trends in automation tooling.

1.4.2 Chapter 3: The Impact of Foundational Models on Patient-Centric e-Health Systems

Chapter 3, shifts focus from open-source platforms to mobile health applications distributed through the Google Play Store, a semi-closed and user-facing SECO to address **Objective 2** and **Objective 3**. It examines how functionalities, primarily AI and FM-related features, have been integrated into real-world e-Health applications, with a focus on:

- Identifying which features are currently in use and how they are described.
- Classifying apps by AI maturity levels using the Gartner AI Maturity Model [68].
- Assessing how app characteristics (e.g., popularity, update frequency, user ratings) correlate with different maturity levels.

To accomplish this, we analyzed 116 patient-centric healthcare applications by automatically extracting and clustering functionalities from app descriptions using a fine-tuned LLM. The feature extraction pipeline closely followed the approach presented in Chapter 2, with domain-specific modifications tailored to mobile health applications.

The extracted features were then categorized according to a five-level AI maturity model (awareness, active, operational, systematic, and transformational) to assess the degree of AI integration within each application. We conducted a quantitative analysis to establish a current snapshot of the landscape in patient-centric e-Health, examining the distribution of AI maturity levels across the dataset. Additionally, we investigated correlations between functional maturity and app popularity metrics, such as download counts and user ratings, to explore how advanced AI capabilities may influence user engagement and market performance. This chapter offers a foundational understanding of how AI, particularly FMs, is being integrated into real-world healthcare apps.

1.4.3 Chapter 4: Conclusions and Future Work

In this final chapter, findings from both empirical studies are brought together to address broader questions around software ecosystem evolution, functionality reuse, and innovation diffusion across different platform types. Key contributions include:

- A comparative discussion of developer-driven (GitHub) vs. user-facing (Google Play) ecosystems.
- A reflection on functional redundancy and how it manifests differently based on user expertise, platform openness, and reuse practices.
- Insights into the implications of AI adoption, particularly in domains where trust, privacy, and usability are essential (e.g., healthcare).
- Practical takeaways for developers, platform managers, and researchers, including how to design to reduce redundancy and foster sustainable ecosystem growth.

The chapter concludes with a discussion of directions for future research, including potential extensions into real-time functionality tracking, user feedback analysis, and deeper explorations of socio-technical co-evolution in SECOs.

Chapter 2

Chapter Two: Extension Decisions in Open Source Software Ecosystem

Abstract

GitHub Marketplace, as an open source software platform, is rapidly evolving, with an average annual increase of 41.23% in new tools by new or existing providers. In such a dynamic SECO, making informed decisions about market entry and product extension is critical for success and requires careful consideration. GitHub Marketplace offers an environment where developers can openly share tools for automating workflows in open source software. However, our preliminary analysis of this ecosystem revealed a significant functional redundancy among the provided tools in this SECO.
To address **Objective 1** and **Objective 3** of this thesis, we employed a graph-based approach to analyze the functional relationships between 6,983 CI Actions and 3,869 providers. We investigated the birth of functionalities and their evolutionary trajectory over time by mining the version control history and different releases of the actions, and identified early contributors. Understanding the dynamics of the GitHub Marketplace is crucial for developers and organizations aiming to innovate and remain competitive in this rapidly evolving environment. By identifying patterns of functional redundancy and tracing the evolution of tools, we provide valuable insights that can inform strategic

decisions on market entry and product development. Our analysis of early contributors and the evolutionary trajectory of functionalities offers developers a roadmap for anticipating future trends and aligning their products with emerging needs.

2.1 Introduction

SECOs drive innovation by fostering collaboration, competition, and reuse [11, 12, 4]. GitHub Marketplace exemplifies this by enabling developers to share, refine, and compete with automation tools, particularly GitHub Actions [5]. As a dynamic marketplace, it reduces development costs and accelerates software evolution through open source contributions [72, 73].

In software engineering, mobile app stores, such as Apple's App Store and Google Play [74, 7], are among the most extensively studied two-sided platforms. While both GitHub Marketplace and mobile app stores function as two-sided platforms connecting developers with users, they differ significantly in openness and user expertise [75, 76]. Mobile app stores, such as Apple's App Store and Google Play, primarily cater to end-users who typically lack technical backgrounds [10, 77]. These platforms distribute compiled applications without providing access to the source code, limiting users' ability to modify or understand the underlying software. In contrast, GitHub Marketplace serves a developer-centric community, offering tools and integrations with accessible source code, fostering collaboration and continuous improvement within the software development ecosystem. The closed nature of mobile app stores ensures a controlled environment, enhancing security and user experience for non-technical consumers. However, this model can restrict innovation and transparency, as developers must adhere to stringent guidelines and cannot freely modify distributed applications [78, 67]. Conversely, GitHub Marketplace, by hosting open source tools, encourages innovation through shared knowledge and collective problem-solving, benefiting a technically proficient audience.

GitHub Marketplace provides access to source code and developer activity, making

it an ideal environment for studying automation tool innovation [10]. However, a significant overlap exists among tools, particularly within categories like CI, where multiple Actions offer similar functionalities. Understanding why developers create new but functionally redundant Actions instead of reusing existing ones is crucial for analyzing innovation patterns, feature evolution, and competition [5, 14].

2.1.1 What is GitHub Action?

GitHub Actions, originally designed for CI/CD tasks, have expanded into a broad automation framework, surpassing alternatives like Travis CI [79]. The marketplace has experienced rapid growth, with Actions increasing from 7,878 in April 2021 to 22,194 in March 2024, highlighting the ecosystem's expansion [13]. GitHub Actions can execute multiple tasks, automating development and build activities using a structured `YAML` format. The `YAML` files define job sequences that outline an Action's functionality, inputs, and outputs, specifying how automation tasks are executed. Actions published on the GitHub Marketplace must include an `action.yaml` file in its root directory, specifying its technical requirements, functionalities, and integration steps [14].

A qualitative study [13] found that developers select Actions based on functional and non-functional attributes, as well as their composition within a single Action. Developers integrate these tools by referencing them in workflow files and can further customize them through forking and pull requests. This open source model fosters continuous collaboration and enhancement, reinforcing GitHub Marketplace as a critical hub for software automation and innovation.

The options for developers looking to automate different workflows in their repositories via GitHub Actions are diverse. They can either create custom Actions, reuse existing ones from other project repositories or explore the GitHub Marketplace, where various prebuilt Actions cater to different functionalities. Furthermore, the developers can customize these Actions by forking repositories and submitting pull requests. This open source approach fosters collaboration and continuous enhancement of automation tools on the GitHub Marketplace.

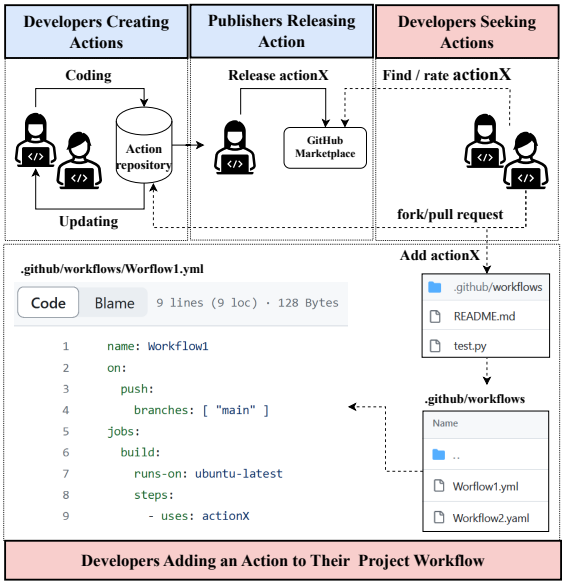

FIGURE 2.1: Actions are developed and shared on GitHub Marketplace, where they are integrated by others to automate workflows.

Figure 2.1 illustrates a simplified scenario, explaining how a publisher releases an Action on the GitHub Marketplace and, subsequently, how users can integrate this Action into their project's workflow to automate tasks within their repository. The process of integrating Actions into repositories is rather a manual effort. Developers can integrate the selected Action into their project by specifying its unique name [14] in the `YAML` file representing their workflow, enabling them to automate workflows within their repositories. To do so, the developer is required to check the `action.yaml` file for integration instructions.

2.1.2 Why Study the GitHub Marketplace?

GitHub Marketplace exemplifies a SECO where diverse automation tools interact [5], making it a valuable case for studying system co-evolution [29, 80]. Its focus on automation (e.g., Actions) highlights how innovation streamlines development, reduces costs, and enhances productivity [81, 82]. The marketplace's open and dynamic nature directly relates to the broader challenges of Systems-of-Systems (SoS) and SECO engineering, positioning it as an ideal case study for analyzing SECO evolution from a Software Engineering perspective. GitHub Marketplace serves as a microcosm of software

development, showcasing how new tools and features emerge, evolve, and compete.

Unlike proprietary platforms, GitHub provides access to source code, usage statistics, and contribution history, enabling empirical analysis of innovation patterns, offering what we refer to as Rich Open Data. This transparency allows researchers to track the development life cycle of automation tools, study adoption trends, and examine the mechanisms that drive feature innovation [15, 16]. Additionally, the marketplace exhibits extensive redundancy and a competitive landscape, where both tool providers and developers possess similar levels of technical literacy. This distinguishes GitHub Marketplace from other software distribution platforms, where users often lack the expertise to assess or modify the software they consume. The availability of full pipeline visibility, including access to source code, enables a unique opportunity for component-based analysis and software reuse, potentially enhancing efficiency and reducing duplication in software development. The presence of overlapping tools raises fundamental questions about competition, redundancy, and differentiation strategies in open source ecosystems [5].

By studying GitHub Marketplace, we gain a comprehensive understanding of SECO evolution, feature emergence, and the interplay between competition and collaboration in driving innovation. These insights establish GitHub Marketplace as a critical platform for SECO research, bridging empirical software engineering with real-world development practices.

To achieve the objectives of this thesis, we analyze the functionalities of GitHub Actions and their relationships in terms of shared features and capabilities (i.e., functional relationships). Additionally, we examine how these relationships have evolved over time, a process we term functional evolution, and identify the competitive factors driving this transformation within the SECO landscape (**Objective 1**). By doing so, we aim to understand how these automation tools and their associated marketplace contribute to innovation in software development (**Objective 3**).

While our analysis centers on GitHub Marketplace, the patterns of functional evolution observed in this study provide broader insights for developers and stakeholders in

other digital marketplaces. The findings contribute to enhanced developer productivity by offering actionable insights that help developers design, refine, and maintain more effective tools while navigating the competitive landscape and complementary products in software marketplaces. Beyond GitHub Actions, our study has far-reaching implications for developers, platform managers, and researchers seeking to foster innovation, reduce redundancy, and drive sustainable growth in other SECOs. To accomplish these goals, we address three main research questions (RQs) in the following section.

2.1.3 Research Questions

We focused on three research questions to evaluate the dynamics of the GitHub marketplace at the functional level:

RQ1: How do the functional relationships among offerings in the GitHub Marketplace evolve within this SECO?

To address this question, we analyzed the `YAML` files of Actions at two time snapshots, t_0 and t_1, to extract and categorize their functionalities. For each point in time, Actions were classified as either independent (having no functional overlap) or overlapping with other Actions. For overlapping Actions, we applied a graph-based approach using `NetworkX` [71] to construct a functional relationship network. This allowed us to visualize and examine the interconnections among Actions, their features, and publishers, offering a structural and topological representation of the GitHub Marketplace ecosystem.

RQ2: How do the provider characteristics relate to the evolution of functional relationships in this SECO?

To investigate this, we examined the characteristics of providers (i.e., publishers) who develop and maintain GitHub Actions, aiming to understand their role in shaping the ecosystem's evolution. We classified providers into two distinct categories: Independent Tool Providers, who exclusively release Independent Actions (i.e., Actions with no functional overlap with others), and Dependent Tool Providers, who release at least one overlapping Action, indicating functional dependency with other Actions. We

then identified Early Contributors, providers who initially (at snapshot t_0) offered an Independent Action that later (at snapshot t_1) developed functional overlap with other Actions. These publishers were classified as Early Contributors to capture their early involvement and influence within the SECO evolution and innovation [83]. By analyzing the relationship between provider characteristics and the evolution of their published Actions, we aimed to determine the role of Early Contributors and innovative providers in driving functional changes and innovation within this SECO.

RQ3: How can the evolution of the functional relationships in GitHub be explained using predefined SECO migration patterns?

To further understand evolutionary trends in the SECO, we examined the migratory behaviors of functionalities to assess the influence of pre-existing patterns on the evolution of Actions in the GitHub Marketplace. Specifically, we leveraged the nine behavioral patterns introduced by Sarro et al. [28] in the context of mobile app stores, where functionalities migrate across app categories, to analyze the extent to which similar migrations occur within the GitHub Marketplace. We mapped the functional evolution observed in the GitHub SECO to these migratory behaviors and empirically examined their role in shaping developer movement, feature evolution, and ecosystem growth. By doing so, we aimed to uncover whether SECO migration patterns observed in mobile ecosystems also apply to software development marketplaces and how these behaviors influence innovation in GitHub Actions.

We begin by summarizing related studies in Section 2.2, followed by an overview of the data collection process in Section 2.3. Next, we present the methodology used in this study in Section 2.4 and discuss our results in Sections 2.5 and 2.6. Finally, we address potential threats to validity in Section 2.7 and conclude with closing remarks in Section 2.8.

2.2 Related Studies

In this section, we summarize the related studies relevant to our topic of discussion. We begin by reviewing research on SECOs, followed by studies on the GitHub Marketplace and GitHub Actions.

2.2.1 Software Ecosystems

SECOs play a crucial role in fostering creativity throughout the software development process [84, 85]. Joshua et al. performed a systematic review of SECOs, highlighting their key features, benefits such as fostering co-innovation and reducing costs, and challenges including security and infrastructure management [80]. Bosch [3] defines a SECO as a commercial ecosystem that includes software solutions and services designed to enable, support, and automate activities and transactions within both social and business environments. In recent years, analyzing SECOs to gain insights into various phases of the Software Development Life Cycle (SDLC) has become an area of growing interest among researchers [73]. Malcher et al. [86] explored Open Innovation (OI) practices used to manage requirements in SECO, where multiple actors interact through various communication channels. They identified 10 OI practices and 14 communication channels, intending to make the requirements management process in SECO more informal, open, and collaborative.

2.2.2 GitHub Ecosystem and Marketplace

Saroar et al. [5] further contributed to the field of SECOs by establishing GitHub Marketplace - an ecosystem for developers to share automation tools, also known as Actions- as a SECO, identifying its components and conceptualizing them within the broader definition of a SECO. Furthermore, they thoroughly analyzed the GitHub Marketplace to bridge the knowledge gap between state-of-the-art and state-of-practice. Their research highlighted that Action providers publish multiple Actions, along with various statistical analyses of different attributes within the GitHub Marketplace. This was the sole study we found pointing to the GitHub Marketplace as an SECO.

2.2.3 GitHub Actions for Workflow Automation

Even though there are limited studies targeting the GitHub Marketplace, various studies focus on GitHub Actions by studying the GitHub repositories that use Actions to automate their workflow [87, 88]. Golzadeh et al. [79] conducted a nine-year empirical study to explore the evolution of CI services within GitHub repositories, discovering that Action has suppressed third-party tools such as Travis. This was further followed from the line of study from the same research group to evaluate the use of Actions for different automated tasks [89, 90]. Khatami et al. [91] later performed a study to identify 22 workflow smells in GitHub Actions by analyzing 10,012 commits and developing automated scripts for detection. Results highlight the need for better contextual understanding and communication with developers. Valenzuela et al. [92] examined GitHub Actions workflow maintenance in 200 projects, highlighting bug fixes and CI/CD improvements as key factors hinting at improved resource planning, best practices, and enhanced tool features. Since then, the field has increasingly focused on studying the automated repositories using these Actions [93, 94] and the impact of using Actions on CI/CD processes [95]. Saroar and Nayebi [13] investigated developers' perceptions of GitHub Actions, focusing on reusability and workflow file creation efficiency, through a survey of 90 developers. They found that the developers opt to create new Actions due to various reasons (e.g., Bugs). Decan et al. [96] conducted a study on the outdatedness of project workflows utilizing GitHub Actions. Their findings revealed that, despite regular updates to Actions, most workflows rely on versions that have been outdated for over seven months.

2.2.4 Feature Extraction & AI for Software

To study the functional evolution of features in the marketplace, we first need to extract features from the tools in the SECO. There have been various methods to extract features from app descriptions in the state of the art. The feature extraction using n-grams and agglomerative clustering by Harman et al. [97] is one of the most popular methods. However, with the rise of generative AI and LLMs in software-related tasks [98, 99,

100, 101, 102], we opted to use LLMs for feature extraction. LLMs have been shown to provide a scalable, explainable alternative to costly human evaluations. Zheng et al. [103] used strong LLMs as judges for open-ended questions, revealing that models like GPT-4 can effectively approximate human preferences with over 80% agreement.

2.2.5 Functional Evolution & Network Analysis in SECOs

To the best of our knowledge, our study is the first to explore the functional evolution of Actions in GitHub Marketplace. However, this is not the first time functional evolution has been studied in SECOs. Sarro et al. [28] performed a study to understand the migratory behavior of the features in a SECO over 33 weeks among pre-defined categories of the SECO. They established nine migratory behaviors and introduced a framework for analyzing app feature lifecycles using data from Samsung and BlackBerry app stores. They found that intransitive features exhibit unique behaviors and highlight correlations between price, rating, and popularity, offering valuable insights for developers. Seidl and Aßmann [104] presented a metamodel to capture and analyze the evolution of variability in SECOs, offering insights into the structure and changes within these ecosystems over time.

Additionally, graphs have been used to study different relations in SECOs. The study by Kim et al. [105], where they explored the content of mobile applications in the App Store using text-mining-based network analysis, is one of such studies. They visualized the associations among categories and applications, presenting both macro-level category networks and micro-level app networks.

2.3 Data collection

Developers can self-categorize their Actions into up to three of the 18 categories available on GitHub Marketplace [66, 14]. These categories represent high-level processes. GitHub Actions were developed initially to automate CI processes, making this category the oldest and most established within the ecosystem. The history of CI-related Actions dates back to 2015, predating the introduction of GitHub Marketplace itself.

```yaml
name: "Action's Name"  ..... # Required/ Must be unique
author: "Action creator"  ..... #Optional
description: "Description of action"..... #Required
input: .....#Optional
    input_ID:..... #Required
        description: "Input description" ..... #Required
        required: "True/False" ..... #Optional
        default: "default value"..... #Optional
        deprecationMessage: "Warning message".....#Optional
output:.....#Optional
    output_ID:..... #Required
        description: "Output description"..... #Required
        value: "Only for composite actions"..... #Required
runs:.....#Required
    using: "runtime used to execute the code"
branding:.....#Optional
```

FIGURE 2.2: A template of `action.yaml` file encompassing all potential fields and sub-fields. For this project, our emphasis was specifically on collecting data from the description fields

Given this longer evolutionary trajectory, CI Actions provide a rich dataset for analyzing functional evolution and competitive dynamics within the SECO. For this study, we scraped data from 6,983 Actions in the "Continuous Integration" category as of January 2024. Valid Actions on GitHub Marketplace require an `action.yaml` file in their root repository [14]; therefore, we excluded any Actions missing this file or lacking a connection to their GitHub repository. Since our study focuses on the evolution of functional relationships among Actions, we retained only those present at both t_0 (no later than October 2023) and t_1 (January 2024), resulting in a final dataset of 5,006 Actions.

We developed an in-house scraper using `BeautifulSoup` [106] to collect data from GitHub Marketplace. For each Action, we extracted metadata, including the Action name, short and long descriptions, publisher name, number of stars, repository link, and other relevant attributes. We then navigated to each Action's repository, cloned it, and retrieved the contents of its `action.yml` file. GitHub Marketplace restricts access to only the first 1,000 Actions per category in its default listing, as noted by Saroar et al. [5]. To circumvent this limitation, we leveraged GitHub Marketplace's sorting algorithms along with a set of search terms to maximize coverage. Specifically, we used bi-grams extracted for each category from Saroar et al. [5], along with a full set of English letters (upper and lowercase) and numbers (0 - 9) to generate search queries. By

combining sorting strategies and search queries, we successfully retrieved 6,983 out of 7,248 Actions (96.34%) in the “Continuous Integration" category, ensuring comprehensive coverage for this study.

Each Action consists of one or more functionalities, which are described in natural language within the Action description and similarly structured in the `action.yml` file, along with its inputs and outputs. The structured format of `action.yml` makes it a more reliable source for feature identification, as it provides a standalone representation of an Action as seen by a developer. A template structure of the `action.yml` file is presented in Figure 2.2. Each `action.yml` file contains seven fields: `name, author, description, input, output, runs, and branding`. Among these, `name, description, and runs` are mandatory, while `author, input, output, and branding` are optional. The `description` field provides an overview of the Action’s functionalities, whereas the `input` and `output` fields define the required inputs and possible outputs of the Action, respectively. Each of these fields also contains its own set of optional and mandatory subfields. Since the description field is mandatory and used to generate the description in the GitHub Marketplace, developers provide detailed explanations of their functionalities within this field. Therefore, we chose to extract all description fields for our analysis.

The data, along with the scripts utilized for mining GitHub Marketplace and Action repositories, in addition to conducting the subsequent analysis, are made publicly available in our GitHub repository [107].

2.4 Methodology

In this section, we outline the methodology employed in this study, organized according to the research question. To study the evolution of Actions, we defined two time points in the version history of Actions: t_0 and t_1, where $t_0 < t_1$. The point t_1 is uniform across all Actions and represents the most recent update of the Action data as of January 2024 (date of last extraction of data from GitHub Marketplace). In contrast, t_0 varies for each Action and marks its initial release, which can be any time between early 2015 to October 2024 (We selected Actions with a minimum time difference of 12 weeks between

Feature Extraction Prompt

Capacity and Role: Requirement Analyst
Insight: We are looking to extract software features and functionalities from software descriptions. software features define the capabilities of the software.
Statement: Extract system functionalities from a given list of descriptions
Personality: accurate and precise and make sure the features are supported by the descriptions.
Experiment: output all features in a Python list. ONLY output the Python list.

here is the descriptions: `{descriptions}`

FIGURE 2.3: Prompt template created by modifying the CRISPE template and used to extract features from Action descriptions.

t_0 - t_1). By assigning a unique t_0 to each Action, we forgo the exploration of the birth of new Actions. However, this approach allows us to remain focused on our primary objective: studying the evolution of functional relationships among existing Actions within the ecosystem.

2.4.1 Extracting Features Overtime (RQ1)

We extracted all available description fields, including the overall `description` and those found in the input and output sections of the `action.yml` file, as the primary source for feature extraction. Subsequently, we identified relationships between these features using a network identification model.

Mining Features from Action Descriptions

From the `YAML` file, we took several steps to extract Action's functionality:

Using LLM to extract features: With the rise of generative AI, particularly LLMs, in supporting Requirements Engineering (RE) tasks [98, 99] and their capabilities as alternatives to human evaluations [103, 108, 109, 110], we opted for LLM-based feature extraction over traditional methods [97]. We evaluated several frameworks on a subset of 100 GitHub Actions and found that the hybrid approach CRISPE [111, 112, 113] produced the most meaningful features. Figure 2.3 illustrates the prompt used to extract

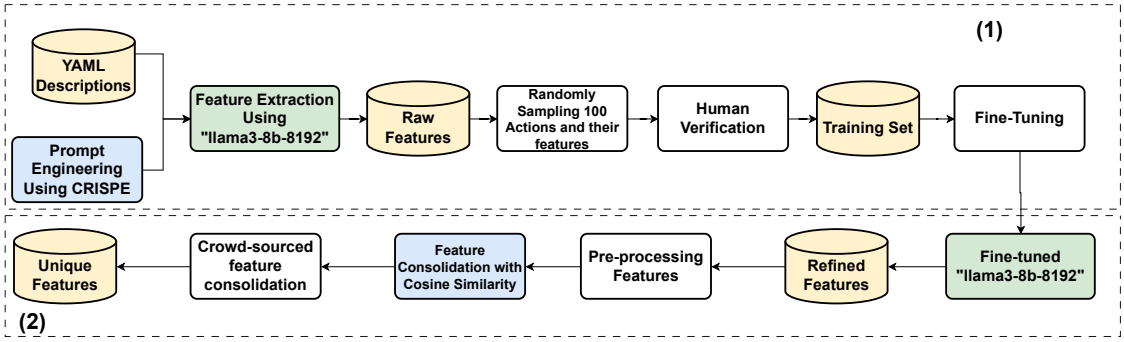

FIGURE 2.4: Feature extraction methodology using LLAMA3-8B-8192 model fine-tuned using manually labeled sample dataset (1). Additionally, we used cosine similarity and crowd-sourcing to consolidate similar features (2)

.

features from our dataset. We tested multiple LLMs, primarily those available through the `Groq` API [114], including MIXTRAL and LLAMA 3. To ensure consistency, we set the temperature to zero. Given the inherent risk of hallucination in LLM-generated outputs, we incorporated human verification to improve accuracy.

Adjusting a subset of features using human annotation: To assess and refine LLM-extracted features, the first author manually reviewed 610 features from 100 randomly selected Actions. They identified inaccuracies in 17.38% of the extracted features, adjusting their mappings to better align with the corresponding descriptions.

Fine-tuning LLM using human annotations: Fine-tuning LLMs and implementing Retrieval-Augmented Generation (RAG) have been shown to enhance output accuracy while significantly reducing model hallucinations in software engineering tasks [115, 116, 117]. Based on the manually corrected features, we fine-tuned the model to improve extraction accuracy. We optimized feature extraction through few-shot prompting and RAG. We designed structured prompts incorporating human-verified feature mappings as exemplars, allowing the model to generalize better across new descriptions.

In our approach, we maintained a repository of manually corrected features and leveraged embedding-based retrieval to fetch the most relevant examples for each input dynamically. This retrieval mechanism ensured that the model was exposed to high-quality, domain-specific feature mappings without requiring parameter updates.

Refining features: We ran the fine-tuned model on the descriptions to refine the

extracted features. In this process, we applied SelfCheckGPT [118], a self-consistency evaluation method that detects potential hallucinations in LLM outputs. This tool assesses Natural Language Inference (NLI), which measures whether a generated feature logically follows from the provided description. A high NLI error rate suggests inconsistencies or hallucinated content, whereas a low NLI rate indicates stronger alignment between the input and output. Using SelfCheckGPT, we found that the LLAMA3-8B-8192 model achieved the lowest NLI error rate (4.96%), meaning only 4.96% of the extracted features were flagged as inconsistent with their original descriptions. This validation confirmed our model selection as it demonstrated the best reliability in feature extraction.

Identifying Unique features: The use of different terminologies among developers may lead to the same feature being expressed in different ways. This inconsistency can complicate subsequent analysis and not reflect the true nature of the ecosystem. Hence, to remove potential duplicates and combine differently worded features, first we preprocessed the features (converting to lowercase, removing stopwords and special characters, and lemmatizing) and then vectorized them with the `bert-base-nli-mean-tokens` model [119] to convert to vector embedding. Then, we used the `cosine_similarity` function of `Scikit_learn` library [120] to calculate the Cosine similarity score among features extracted at both t_0 and t_1 to identify unique features in the entire datasets and established a set of "Unique Features" (Figure 2.4). We used the recommended cosine similarity of 90% [121] to identify high similarity among these features at the two snapshots of the same action and between the Actions. Furthermore, we manually checked the features using crowd-sourcing and combined the similar features as identified by human annotators.

Identifying Network of Functional Relations

During the initial analysis of the features extracted from the marketplace, we identified that these tools (and subsequently their providers) are interconnected via the features

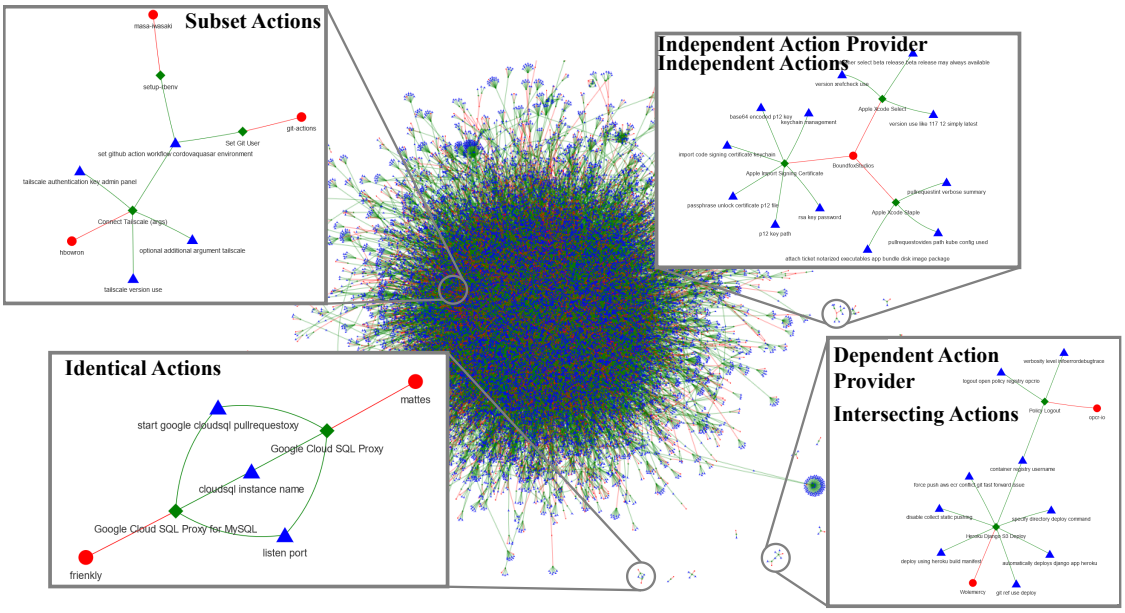

FIGURE 2.5: A highly connected network of Action functionality in the "Continuous Integration" category. The red circles represent each publisher connected to their released Actions (demonstrated in green squares), which are connected to their respective features in blue triangles. We identified the relations discussed in section 2.4.1 and section 2.4.2 and labeled them in the network.

they offer. We refer to these connections as functional relations and argue that these relations can be classified into four distinct types: *Independent*, *Subset*, *Intersect*, and *Identical* (**RQ1**). These relationships are formally defined as follows:

Definition 1 (*Feature Set*). A tool A can be defined as a set of features $f_1, ..., f_n$ that describes its functionalities:

$$A = F_A = \{f_1, f_2, f_3, ..., f_n\}$$

In other words, F_A is the *feature set* of tool A.

Definition 2 (*Independent Tools*). A is an independent tool if for every feature f in feature set F_A, feature f does not belong to any other tool's feature set; in other words, no other tool in the ecosystem has features offered by this tool:

$$A \in A_{independent} \iff \forall f \in F_A | f \notin F'_A$$

where $A_{independent}$ is a set of all independent tools in the ecosystem.

Definition 3 (*Subset Relation*). Consider tool A with feature set F_A and tool B with feature set F_B. We define A is subset of B if all features in F_A are also in F_B:

$$A \subset B \iff \forall f \in F_A | f \in F_B \wedge F_A \neq F_B$$

in other words, tool B has all features of tool A with some additional features.

Definition 4 (*Identical Tools*). These types of tools are a special case of tools with a subset relation, where tool A with feature set F_A and tool B with feature set F_B have identical features:

$$A = B \iff F_A = F_B$$

Definition 5 (*Intersect Relation*). For tool A with feature set F_A and tool B with feature set F_B, we define A intersects with tool B if at least one feature in F_A is also in F_B:

$$\begin{aligned} \mathrm{A} \cap B \iff \exists f \in F_A | \quad & f \in F_B \quad \wedge \\ & F_A \neq F_B \quad \wedge \\ & A \not\subset B \quad \wedge \\ & B \not\subset A \end{aligned}$$

In other words, tools A and B have at least one feature in common but are not a subset of each other nor share the exact same features.

We then employed `NetworkX` and `PyVis` libraries to construct and visualize the GitHub Marketplace ecosystem's network at two distinct time snapshots. `NetworkX` facilitated the creation and analysis of complex networks by allowing us to initialize directed graphs, add nodes representing individual Actions, and define directed edges to denote dependencies or interactions between them. Additionally, we applied various graph algorithms from `NetworkX` to explore key structural properties, including `degree_centrality` [122] for identifying highly central nodes, `connected_components` [123] to uncover clusters of related actions, and to detect patterns of common functionalities across the network.

Furthermore, the scalability of graph analysis tools allows for the efficient examination of large-scale Action networks for larger categories of the marketplace (e.g., CI category). The most current network (at time t_1) is presented in Figure 2.5. The Actions, features, and publishers are depicted as nodes of green squares, blue triangles, and red circles, respectively, with edges connecting Actions sharing identical features and

publishers. We studied this network to identify the relations and entities formalized in section 2.4.1 and section 2.4.2 and any other possible relations.

2.4.2 Understanding the role of Providers in the SECO (RQ2)

To examine how provider characteristics relate to the evolution of functional relationships within the GitHub Marketplace SECO, we conducted an empirical study using a longitudinal analysis [124, 30, 47] of publisher behaviors between two time snapshots, t_0 and t_1. Our methodology consists of three key steps: (i) identifying tool providers and categorizing them as independent or dependent, (ii) tracking their evolution over time, and (iii) analyzing the role of early contributors in shaping the ecosystem.

Using the established network in section 2.4.1, we identified two relationships between the providers and the Actions they release: *Independent Tool Provider* and *Dependent Tool Provider*. We formally define these as follows:

Definition 6 (*Independent Tool Provider*). A provider in the SECO is an independent tool provider if and only if they publish independent tools. Consider provider P publishing a set of tools $A_P = \{A_1, A_2, ..., A_n\}$, P is Independent Provider if every tool in A_P is an independent tool:

$$P \in P_{independent} \iff \forall A \in A_P | A \in A_{independent}$$

where $P_{independent}$ is a set of independent tool providers in the ecosystem.

Definition 7 (*Dependent Tool Provider*). A tool provider in the ecosystem is a dependent tool provider if it publishes at least one tool that is not independent. These providers publish tools that are either a subset of or intersect with other tools. Consider provider P who has published a set of tools $A_P = \{A_1, A_2, ..., A_n\}$, P is dependent provider if there is one tool in A_P that is not an independent tool:

$$P \in P_{dependent} \iff \exists A \in A_P | A \notin A_{independent}$$

where $P_{dependent}$ is a set of dependent tool providers in the ecosystem.

To assess how provider relationships evolved, we tracked the functional status of each provider from t_0 to t_1. This analysis focused on how previously independent tool

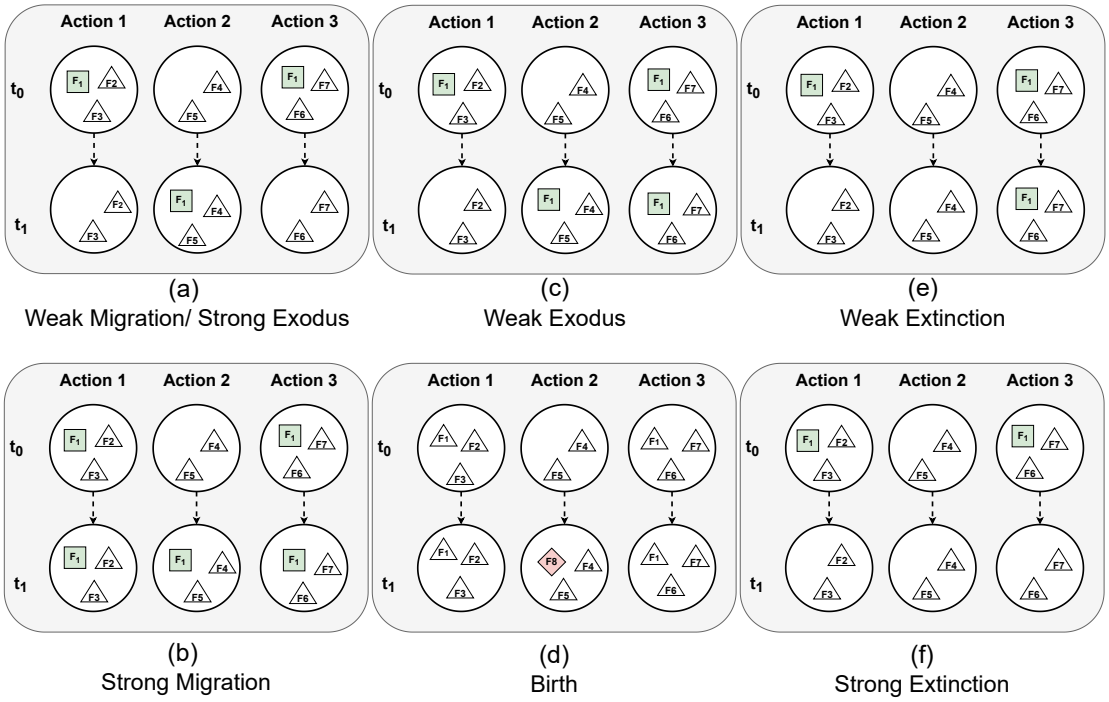

FIGURE 2.6: Modified Migratory behavior by Sarro et al. [28] where features migrate among tools **(RQ3)**.

providers transitioned to dependent providers by incorporating or intersecting with existing functionalities. In the network model, the providers are represented as distinct nodes (depicted by red circles), each potentially surrounded by multiple Actions they have published.

2.4.3 Formalized Migratory Behavior in GitHub Marketplace (RQ3)

To address **RQ3**, we adapted the migratory behaviors outlined by Sarro et al. [28] to focus on inter-category migrations rather than the intra-category migrations used in their original study. Sarro et al. [28] identified nine migratory behaviors for features across different categories at two time points, t_0 and t_1, separated by at least 33 weeks. We selected eight of these behaviors and redefined them to track migration across different Actions, as opposed to categories. This adjustment shifts the focus from inter-category to intra-category migration. We opted to use eight out of nine of these migratory behaviors since one of these behaviors (i.e."Unborn") is not in the context of our study.

In the following section, we provide a summary of these eight behaviors and present a visualization in Figure 2.6.

Weak Migration. A feature has a weak migration if it was present in at least one tool at t_0 and migrated to at least one new tool at t_1. The features may disappear from one or all tools they resided in at time t_0 (Figure 2.6 - a).

Strong Migration. Similar to weak migration, a feature has a Strong migration if it was present in at least one tool at t_0 and migrated to at least one new tool at t_1 with the difference that the features remain with all the tools it resided in at t_0 (Figure 2.6 - b).

Weak Exodus. This behavior is a special case of weak migration where a feature at t_1 disappears from at least one tool it resided in at t_0 and appears in at least one new tool at time t_1 (Figure 2.6 - c).

Strong Exodus. This behavior is another special case of weak migration where a feature at t_1 disappears from all tools it resided in at t_0 and appears in at least one new tool at time t_1 (Figure 2.6 - a). While all instances of strong exodus are cases of weak migration, the converse does not hold.

Birth. This behavior is a special case of Strong Exodus where the feature does not exist in any tools at t_0 and appears in one tool at t_1 (Figure 2.6 - d).

Intransitive. In this type of behavior, the feature at t_1 does not disappear from any tool it previously resided in, nor does it appear in any new tool.

Weak Extinction. A feature experiences weak extinction if it disappears from at least one tool at t_1 and does not appear in any new tool (Figure 2.6 - e).

Strong Extinction Similar to weak extinction, the feature is said to have strong extinction if it does not appear in any new tool at t_1 and while disappearing from all tools (Figure 2.6 - f).

Sarro et al. [28] also defined undead or unborn features as a special case of strong extinction to be features that are outside of the ecosystem. However, we have not observed such relations in the GitHub marketplace. These defined migratory behaviors in GitHub Marketplace follow the same subsumption hierarchy as in the original work[28].

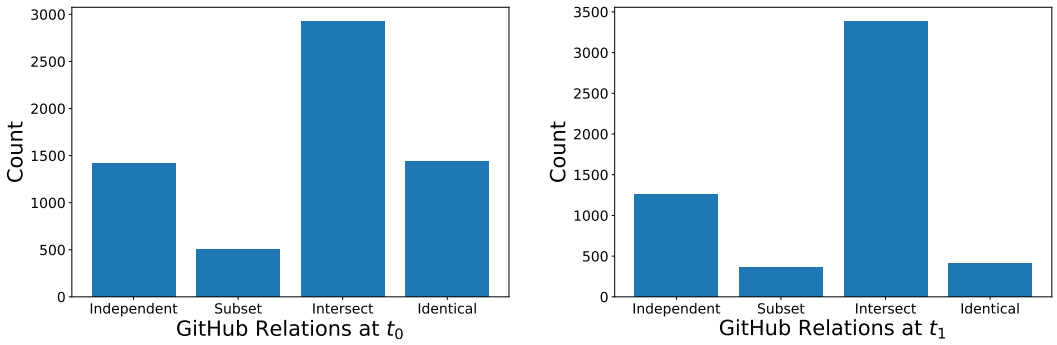

FIGURE 2.7: Count and distribution of relations at t_0 (a) and t_1 (b). Over time, the number of Actions that are subsets of other Actions and the number of Identical Actions have decreased, whereas the counts of Independent and Intersecting Actions have remained relatively stable with a slight increase (a)

2.5 Empirical Results

In this section, we will examine and discuss our results from analyzing 10,694 features at time t_0 and 16,537 features at t_1 across 5,006 Actions in the “Continuous Integration" category of GitHub Marketplace gathered in section 2.4 to answer our three research questions.

2.5.1 Functional Evolution in GitHub Marketplace (RQ1)

To address this research question, we extracted functionalities from `YAML` descriptions, constructed a network using these functionalities, and identified relationships among Actions. In the following, we present our results for **RQ1**.

Mining Functionalities from Action Descriptions

To analyze the functionalities of Actions, we extracted features from the "Continuous Integration" category using a combination of manual annotations and LLMs. Specifically, we focused on extracting functional descriptions from `action.yml` files, as they provide structured information about each Action’s capabilities.

Overall, we extracted 21,477 raw features at time t_0 and 38,583 raw features at time t_1, requiring further refinement. We refined the extracted features using fine tuned language model, which reduced the feature set from 21,477 to 13,994 (34.84% reduction) at t_0 and from 38,583 to 20,569 (46.69% reduction) at t_1. Further manual refinements through crowd-sourcing resulted in 10,694 unique features at t_0 and 16,537 unique features at t_1, which we used for our functional analysis.

Identifying Network of Functional Relations

At t_0, we identified 10,694 unique features, with Actions containing between one and 104 features (an average of 3.42 features per Action). Additionally, we identified 1,186 disjoint components, where nodes are interconnected within a component but not connected to any nodes outside of it. Following the definitions provided in Section 2.4.1, we categorized Actions into 1,417 independent Actions, 503 subset Actions (i.e., a strict subset of another Action), 1,438 Actions that are identical to at least another Action, and 2,929 intersecting Actions (partially overlapping functionalities). Figure 2.7.a-1 shows these distributions of these relations with details.

At t_1, the ecosystem had grown, containing 16,537 unique features compared to t_0, with each Action possessing between one and 78 features (mean: 4.93 features per Action). We identified 1,033 disjoint components, suggesting an increasingly interconnected ecosystem. The distribution of Actions evolved as 1,261 independent Actions, 364 subset Actions, 410 identical Actions, and 3,387 intersecting Actions, which shows a notable decrease in the number of subset and identical Actions, alongside an increase in intersecting Actions, suggesting a shift towards greater feature integration and synergy among Actions. We depicted these distributions in Figure 2.7.a-2.

Over time, the GitHub Marketplace SECO has undergone significant structural changes, with increasing feature integration, reduction in redundancy, and growth in complexity. The overall number of features increased by 54.64%, while the number of disjoint functional components decreased by 12.90%, indicating a shift toward a more interconnected and cohesive ecosystem. Between t_0 and t_1, we observed an 11.01% reduction

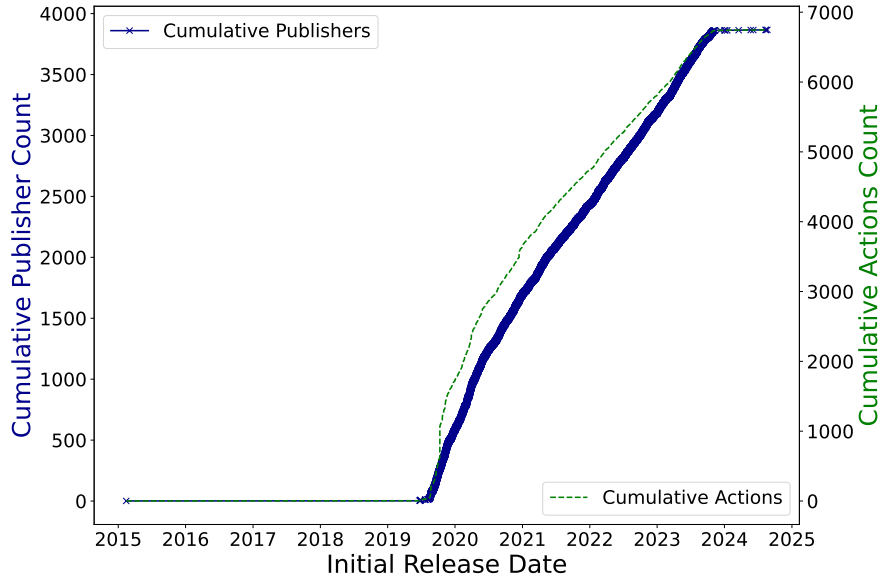

FIGURE 2.8: Number of publishers releasing the initial version of their Action over time. As time progressed, more new publishers and Actions joined the marketplace, whereas 85 publishers have been producing Actions ahead of their competition since 2015 **(RQ2)**.

in independent Actions, alongside a 15.64% increase in intersecting Actions. This trend suggests that developers are building on existing functionalities rather than creating entirely distinct Actions, contributing to a more interdependent SECO. Additionally, the average number of features per Action increased by 44.15%, reflecting growing functional complexity over time.

Interestingly, the number of subset Actions decreased by 27.63%, and identical Actions dropped sharply by 71.49%, indicating a gradual streamlining of redundant functionalities. This suggests a natural selection process, where functionally identical tools are either absorbed into broader offerings or evolve into more distinct, specialized tools.

Moreover, we found that Actions are not limited to a single relationship type -many exhibit multiple forms of interconnection, such as being both a subset and an intersecting Action simultaneously. These "co-occurring relations" contribute to a more intricate network, where Actions evolve dynamically within the SECO (Figure 2.5).

2.5.2 Providers' Role in Functional Relations (RQ2)

We define t_0 as the time at which each Action in our dataset is first introduced into the ecosystem. In total, we identified 3,867 publishers in the ecosystem at this time (first release of each Action). For this analysis, we ignore the birth and extinction of individual Actions and providers over time. In Section 2.4.2, we introduced two key provider-related terms: *Independent Tool Providers* and *Dependent Tool Providers*. Independent tool providers are those in the SECO who exclusively publish independent Actions - that is, all of their Actions are unique in terms of features and functionality. In contrast, dependent tool providers publish at least one Action that is not entirely unique but is interconnected with other Actions, either by being identical to, a subset of, or intersecting with them. At the time t_0, 1,043 of 3,867 publishers were classified as independent tool providers (26.97% independent and 73.03% dependent tool providers). However, by t_1, this number had decreased to 927, reflecting a 11.12% decline over time. This suggests that, over time, more publishers are relying on existing features within the ecosystem to create and maintain their Actions, with fewer publishers remaining completely independent in their offerings. This trend may indicate a consolidation within the GitHub Marketplace, where the number of distinct tools diminishes and more interconnected tools emerge, hinting at growing competition and/or collaboration among providers. The slight increase in dependent tool providers from 2,824 at t_0 to 2,940 at t_1 further supports this notion of elevated competition and collaboration, as more publishers are leveraging pre-existing features in the ecosystem.

Early Contributors in GitHub Marketplace

We found that even though the GitHub Marketplace was introduced in 2019, some publishers have been releasing Actions ahead of their competition from early 2015 to September 2019 (Figure 2.8). We therefore define these publishers who have released their Action before 2020 as "early contributors". We identified 85 early contributors in our ecosystem, releasing 90 actions between them. Among these, only four publishers release two or more actions before the rest of the competition. These publishers are

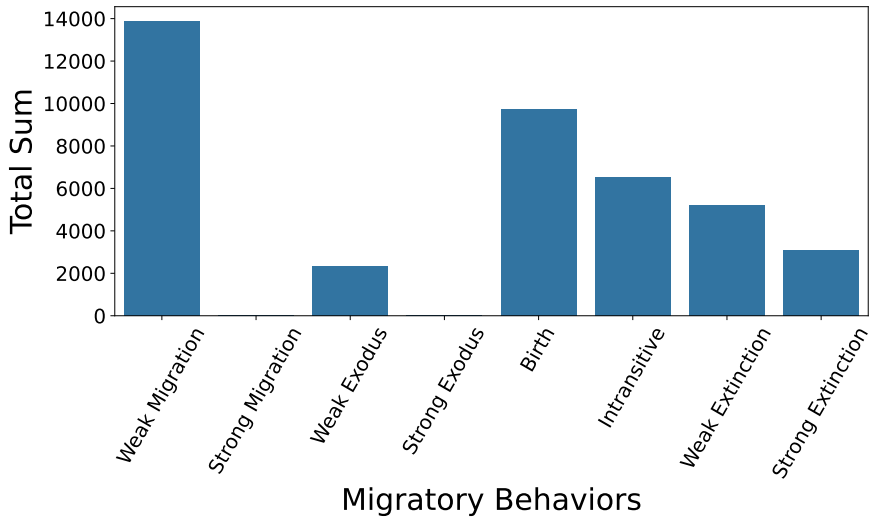

FIGURE 2.9: Total sum of features demonstrating each of the eight migratory behaviors **(RQ3)**.

“Azure", “garygrossgarten", “varunsridharan", and “warrenbuckley", None of which are subsets of any other Actions at t_0. Among these publishers, “varunsridharan" is the only one producing one independent Actions while its other Action intersect with 13 other Actions at t_1. On the other hand, the two Actions released by “Azur" intersect 11 times with other Actions while 38 Actions have identical features as those in two Actions released by “warrenbuckley". This shows that many actions have, in fact, used the features offered by these four as a template to build their products.

We found eight independent tool providers among early contributors (all releasing one Action except one) out of which half remain independent at t_1 while the rest transform into dependent tool providers.Furthermore, we identified 77 early contributors that are also dependent tool providers at t_0. The majority of these contributors release only one Action, with four contributors releasing two or more Actions. Among these contributors, seven transform into independent tool providers.

In summary, our analysis reveals a significant evolution in the ecosystem from t_0 to t_1. Initially, a substantial portion of publishers operated as independent tool providers, but over time, the landscape shifted towards a greater reliance on interconnected features, evidenced by the decline in independent offerings and the increase in dependent

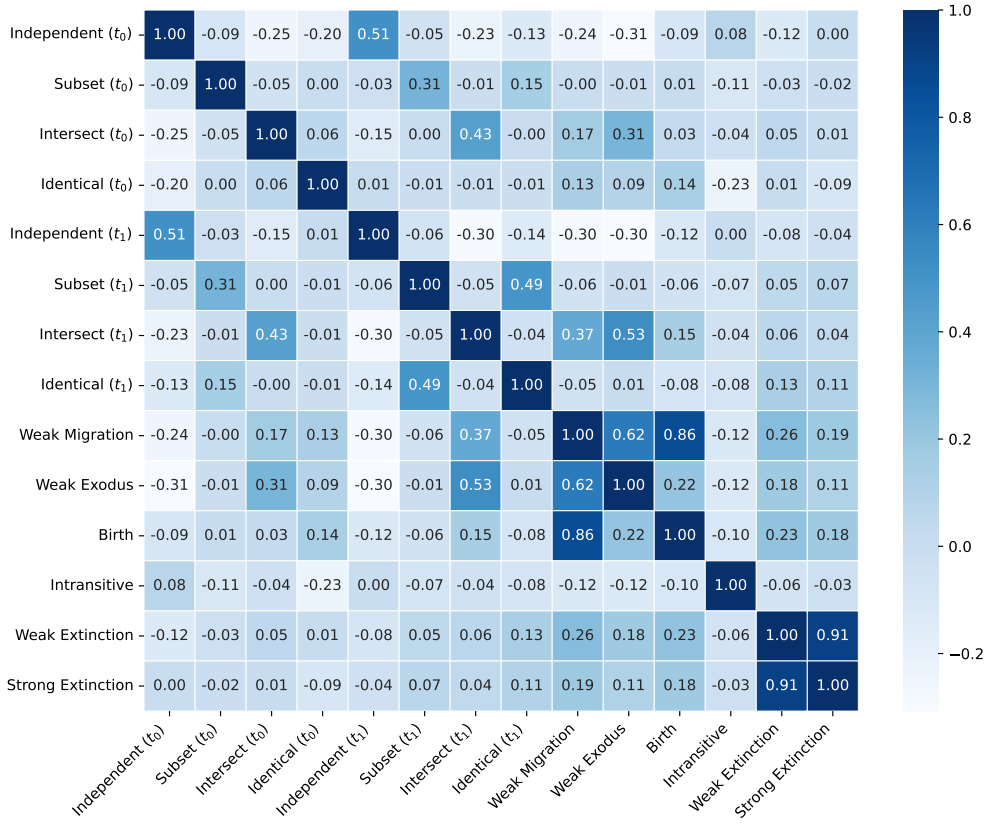

FIGURE 2.10: Correlation among functional relations and migratory behaviors in the SECO **(RQ3)**.

tool providers. This transition suggests a trend towards greater collaboration and competition within the GitHub Marketplace, with publishers increasingly integrating and building upon existing features. The identification of “early contributors", who released Actions before the official marketplace introduction, underscores the role of early innovation in shaping the current ecosystem. Notably, while some early contributors remained independent, many others have transitioned towards a model of intersecting with other Actions to some degree, indicating that the Actions of these early contributors have become a building block for other Actions. Thus, the results suggest that as the marketplace matures, the increasing interplay between independent and dependent tool offerings reflects a shift towards a more interconnected and dynamic ecosystem. This reshaping of the competitive and collaborative landscape aligns with the observed patterns of integration and adaptation among publishers.

2.5.3 Migratory Behavior of Actions in the SECO (RQ3)

To address this research question, we identified a total of 19,099 unique features within the SECO. For each feature, we analyzed its presence at two time points, t_0 and t_1, to classify its migratory behavior according to the eight types defined in Section 2.4.3. Figure 2.9 illustrates the frequency of each migratory behavior observed among the features in the ecosystem. Our analysis shows that Weak Migration is the most common behavior, followed by Birth, Intransitive, Weak Extinction, Strong Extinction, and Weak Exodus. We did not observe any Strong Migration or Strong Exodus. This pattern indicates that migration and growth (Birth) are the dominant dynamics within the SECO. In contrast, intransitive behaviors and extinctions occur frequently but are not as dominant, suggesting that the SECO is in a growth phase with new features and products continually entering the market and existing ones being modified.

We also calculated the correlation between the functional relationships discovered in the SECO and the migratory behaviors (Figure 2.10). Our findings reveal a high correlation among migratory behaviors (likely due to their hierarchical relationships [28]). Additionally, there is a moderate correlation between an Action being independent at t_0 and remaining independent at t_1, as well as between an Action having intersections at t_0 and also having intersections at t_1. Actions with features exhibiting Weak Migration or Weak Exodus behaviors are more likely to evolve to intersect with other Actions and less likely to remain independent.

Furthermore, we analyzed the distribution of each migratory behavior across all Actions (Figure 2.11). Our findings show that every Action contains at least one feature exhibiting one of these migratory behaviors. Specifically, the behaviors associated with growth, weak migration and birth, are the most dominant, with the highest average occurrences per Action. In contrast, the absence of Strong Migration and Strong Exodus indicates that features are highly dynamic, rarely remaining within or completely disappearing from their initial host.

In summary, our analysis of the SECO reveals that migration and growth behaviors are dominant, reflecting an ecosystem that is actively evolving with new features and

products entering the market. The Dominance of Weak Migration and Birth behaviors demonstrates the dynamic nature of the ecosystem, where features frequently transition to new Actions and new features emerge regularly. Meanwhile, the absence of Strong Migration and Strong Exodus behaviors suggests that while features are highly mobile, they rarely remain with or entirely leave their initial host. The moderate correlations observed between functional relationships and migratory behaviors further indicate that Actions with certain characteristics are more likely to exhibit specific types of migration. These insights provide a deeper understanding of the evolutionary processes within the SECO and highlight areas for future research to explore how these dynamics might influence the long-term stability and growth of the ecosystem.

2.6 Discussion & Future Works

GitHub Marketplace is an example of SECOs for developers to share their automation solutions in the form of Actions. Since its release in 2019, the marketplace has experienced an exponential increase in the number of Action and providers. Even though there have been many studies concerning Actions and their use in GitHub-hosted projects, very few studies have targeted the GitHub marketplace itself. We performed an evolutionary analysis of the Actions in the marketplace to provide Action developers with knowledge of how their product might evolve and how they can employ these evolutionary patterns to maintain a successful product.

2.6.1 Evolutionary Analysis of Actions

In this study, we focused on “Continuous Integration" Actions available on the GitHub Marketplace. We identified 5,006 Actions for our analysis and extracted a total of 19,099 features at two time points in the version history of the Actions. We noted that the Actions have become more complex over time, with a higher average number of features per Action. Additionally, the Actions have been shown to evolve to have higher synergy with less number of independent Actions. Furthermore, this synergy is rarely the case

where the Actions are a subset of or identical to each other and happen when their features tend to overlap. We also found that independent Actions are more likely to retain their status than other types of Actions, even though the number of such Actions is declining. Independent Actions represent the most novel form of creativity in the SECO, where the features are unique and uncommon. Hence, a decrease in independent Actions can be a sign that the existing Actions contain most of the required features by the developers, and creating an optimal workflow for automation is a matter of finding the "right" combination of Actions. Furthermore, we found that there are Actions in the SECO that are used as the foundation for other actions, where developers modify existing Actions by adding additional features. Even though we did not consider feature content and characteristics in this study, future work could explore these aspects in detail to uncover how specific feature types, such as performance optimizations or integration functionalities, drive ecosystem dynamics and user adoption. This could complement our findings and provide a deeper understanding of the role of feature evolution in shaping software ecosystems

2.6.2 Super Action Concept

Considering the fact that the set of automation goals is finite and the existence of many Actions with overlapping features raises an interesting question regarding the value of developing a single comprehensive "Super Action" that automates multiple tasks simultaneously versus using independent Actions as modular building blocks to build highly customized workflows. In Figure 2.12, we illustrate a conceptual "Super Action", where the value of each feature (represented by bubble size) must strike a balance with the dependencies and synergies among co-occurring features. To better tailor and customize Actions for repository automation, it is crucial to develop methods for quantifying the synergy among tasks automated by GitHub Actions. Additionally, investigating the value and synergy inherent in these tasks is essential. Constructing the feature set for such Super Actions can be approached as a search problem, similar to the techniques used in release planning by Nayebi et al. [52]. Answering these challenges and their

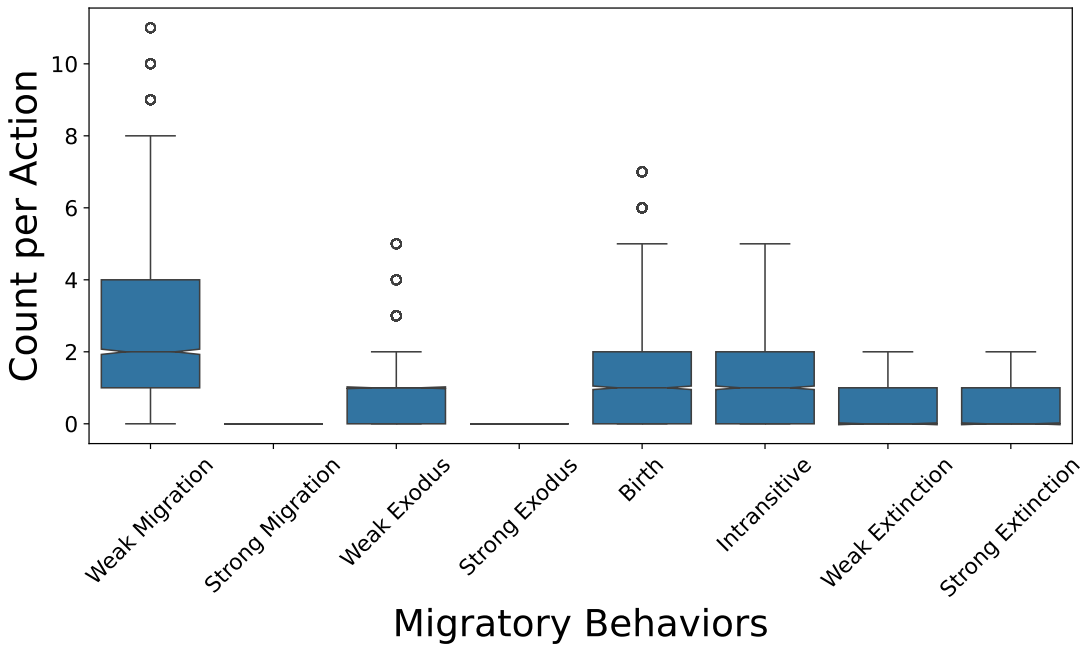

FIGURE 2.11: Distribution of migratory behavior in the Actions in SECO **(RQ3)**.

implications is left for future studies where one can explore whether creating a Super Action offers strategic advantages or if developers prefer to build their workflows using modular components like independent Actions. Overall, we found that the functional relations in the GitHub Marketplace SECO have evolved into a complex and intricate network where many Actions share common features.

2.6.3 Provider Patterns and early contributors

We extended our analysis by examining the influence of provider patterns on the evolution of functional relationships within the SECO. Our dataset revealed 3,869 providers, categorized into independent and dependent tool providers. Over time, the evolution of these functional relationships indicated a notable shift towards dependency, with more providers relying on existing features to maintain their products rather than creating novel, independent tools. Additionally, we investigated the role of early contributors in driving this evolutionary process. We identified 85 early contributors, whose Actions

exhibited significant overlap with other Actions over time, manifesting as subsets, intersecting, or identical relationships. These overlaps suggest that the feature sets developed by early contributors have become foundational for other providers, influencing the creation of new tools. The degree to which these feature sets are “good enough" or require further enhancement, as seen in SECO’s evolution, remains an open question for future research.

2.6.4 Migratory Patterns of Features

We also explored the functional evolution of features through the lens of predefined migratory patterns as described by Sarro et al. [28]. By analyzing 19,099 features at two points in the version history of Actions within SECO, we observed that features have evolved to exhibit behaviors consistent with growth and migration across Actions. Additionally, certain migratory behaviors, such as weak migration, exodus, and birth, were found to increase the likelihood of feature intersections among Actions. These behaviors suggest that features are not only evolving but also being redistributed or repurposed, reinforcing functional overlaps within the ecosystem. Future studies could delve deeper into the dynamics of these migratory behaviors by examining their long-term effects on tool innovation and diversity. Specifically, investigating whether features associated with exodus or weak migration signal a decline in originality, or whether they act as stepping stones for functional synergy, could provide insights into how the SECO evolves to either foster or limit innovation.

In conclusion, our study provides a comprehensive evolutionary analysis of GitHub Marketplace’s SECO, specifically focusing on “Continuous Integration" Actions. The findings highlight the increasing complexity of Actions, a growing reliance on shared features, and a shift towards dependent tools over time. Early contributors play a crucial role in shaping the ecosystem, with their feature sets serving as a foundation for future development. The migratory patterns of features further demonstrate how functionalities are repurposed, contributing to the ecosystem’s dynamic nature. Future research should investigate the long-term impact of these patterns on innovation, particularly

whether they drive or constrain the diversity of tools and features within the ecosystem.

2.7 Threats to Validity

In this section, we will examine threats to the validity of our study and results. During the study, we took precautions to ensure we were limiting the threats to the validity of results however, our results are still under the influence of the following threats:

Internal Validity concerns whether the observed changes in our dataset over time can be attributed to genuine shifts in the ecosystem rather than confounding factors. We aimed to ensure internal validity by employing consistent methodologies for feature extraction and network analysis across both time points. The significant increase in the number of features and the decrease in disjoint components suggest substantial changes in the ecosystem's interconnectedness, supporting the validity of our findings.

Despite our efforts, there are potential threats to internal validity. We utilized LLMs for feature extraction, which, while leveraging state-of-the-art techniques to minimize hallucinations and enhance accuracy, may still introduce errors due to the inherent limitations of these models. Although we manually labeled a small sample to fine-tune the model and consolidated the extracted features using cosine similarity and crowdsourcing, some inaccuracies may persist, potentially affecting our results. Additionally, the process of Action extraction involved using keywords, sorting algorithms, and web crawlers. Even though we managed to extract 96.34% of actions in the SECO, there is a possibility that some Actions were overlooked or not fully captured. This could lead to an incomplete dataset, which might influence the accuracy and completeness of our analysis. Thus, while our methodology strives to ensure robust internal validity, these potential issues should be acknowledged and considered when interpreting our findings.

Construct Validity ensures that our measures and analyses accurately capture the theoretical constructs we intend to study. In this analysis, we defined and categorized actions based on their feature relationships (e.g., independent, subset, intersecting) and

tracked changes over time. Our use of network analysis to identify and categorize these relationships aligns with the theoretical constructs of feature interconnectedness and ecosystem complexity. However, the accuracy of these constructs relies on our theoretical definitions and the quality of our feature extraction. Future work could further validate these constructs by comparing them with other established metrics of ecosystem complexity and feature integration. Our reliance on the descriptions provided in the `action.yaml` file, which depends on developers accurately recording the functionalities of Actions, presents another risk. In some cases, developers may not have fully or correctly documented the functionalities, leading to discrepancies between the actual and recorded features of the Actions.

External Validity pertains to the generalizability of our findings beyond the specific dataset and time frame analyzed. Our study focuses on the GitHub Marketplace's "Continuous Integration" category, which may not fully capture the dynamics of other categories or comparable ecosystems. Although the observed trends - such as increased feature overlap and reduced redundancy - offer valuable insights into this specific domain, caution should be taken when extending these conclusions to other categories or platforms. However, the identified relations and the methodologies themselves can be used for any SECO with minimal modification. To enhance the generalizability of our results, future research could explore whether similar patterns emerge across different categories or software ecosystems.

Conclusion Validity evaluates whether our statistical analyses support the conclusions drawn about the relationships and trends observed in the data. The observed increase in feature counts, changes in action relationships, and the evolution of independent and intersecting actions are based on rigorous statistical analyses of the dataset. The significant reductions in redundant and independent actions, along with the increase in feature overlaps, support our conclusions about the growing interconnectedness of the ecosystem. Nevertheless, ensuring the robustness of these conclusions requires consideration of potential limitations such as sample size, feature extraction, and the accuracy of our theoretical definitions. First, while our sample size is substantial,

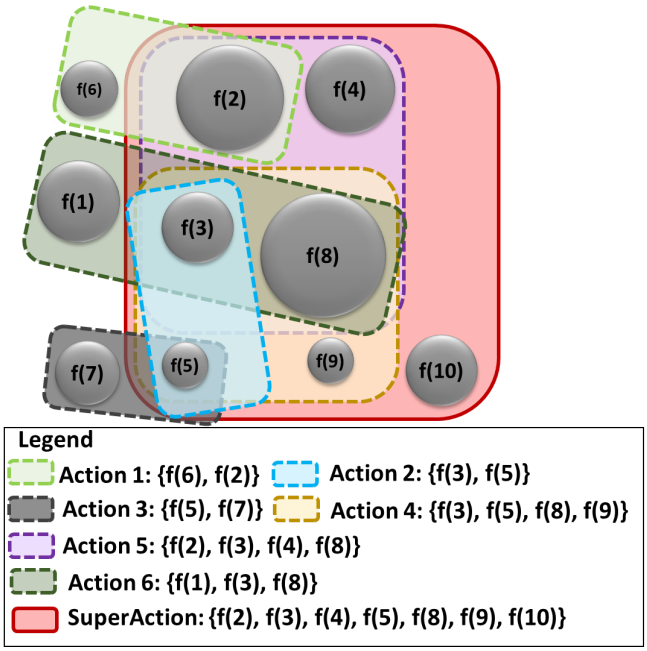

FIGURE 2.12: An example of the value of features and their co-occurrences and the concept of SuperActions.

future research could benefit from larger and more diverse samples to enhance generalizability. Despite our efforts to maintain the consistency of extracted features, there could still be inaccuracies in feature identification and categorization. Additionally, the accuracy of our theoretical definitions is critical. Furthermore, we have assumed that the Action developers use the `action.yaml` as a technical interface and accurately record the functionalities offered by their actions in this document. However, any model limitations or assumptions might influence the observed trends. Addressing these factors through additional validation and refinement of our methods will be crucial for ensuring the robustness and reliability of our conclusions.

2.8 Conclusion

GitHub Marketplace is a unique source of data in the sense that it allows access to both open source data and developer activities. Utilizing this platform, we gathered 6,983 Actions from GitHub Marketplace to understand how the automation tasks are defined and connected among Actions in the SECO. Through our mining and graph analysis,

we identified independent and overlapping Actions and found that Actions evolve to have more intricate interconnected functional relations. We identified 3,869 publishers and discovered that developers often use Actions created by early contributors as foundational elements for their products. Additionally, we analyzed 20,898 features and categorized them according to their migratory behavior. Our findings reveal high feature birth and migratory behavior (e.g., Weak Migration), indicating that features not only evolve but are also repurposed within Actions. This leads to an increasing interconnectedness within the SECO. In future studies, we aim to investigate the possibility of creating “Super Action" further by developing methods to measure the value and synergy among automated tasks within GitHub Actions, while evaluating the benefits of creating “Super Actions" versus customized workflows built from independent Actions.

Chapter 3

Chapter Three: The Impact of Foundational Models on Patient-Centric e-Health Systems

Abstract

As AI becomes increasingly embedded in healthcare technologies, understanding the maturity of AI in patient-centric applications is critical for evaluating its trustworthiness, transparency, and real-world impact. In this chapter, to address **Objective 2** and **Objective 3** of this thesis, we investigate the functional maturity, specifically integration and maturity of AI feature in 116 patient-centric healthcare applications. Using LLMs, we extracted key functional features, which are then categorized into different stages of the Gartner AI maturity model. Our results show that over 86.21% of applications remain at the early stages of AI integration, while only 13.79% demonstrate advanced AI integration.

3.1 Introduction

AI is rapidly gaining traction across various sectors, including health care. However, the current state and maturity of its integration into real-world mobile health applications remain largely underexplored. In particular, it is not yet clear who the primary users of these AI-enabled features are, patients or health care providers, and for what specific purposes they are being employed. FMs, large-scale AI models trained on diverse and extensive datasets, have recently emerged as a transformative force across multiple domains. These models, including LLMs, vision models, and multimodal systems, are increasingly integrated into the sociotechnical landscape due to their capacity to generalize tasks with minimal fine-tuning. Among the many sectors exploring their potential, health care stands out for its growing efforts to harness FMs to support clinical decision-making, streamline workflows, and enhance the quality of patient care.

Within this context, Health Foundation Models (HFMs) have emerged as a specialized subclass of FMs trained on biomedical literature, Electronic Medical Records (EMRs), clinical notes, and other health-related datasets [23]. These models are designed to enable a wide range of applications, from predictive diagnostics and personalized treatment recommendations to natural language interfaces for clinical systems. HFMs are particularly promising for advancing patient-centric e-Health solutions, where AI-driven tools aim to empower patients, increase engagement, and support the delivery of tailored, high-quality care.

Despite these opportunities, integrating HFMs into real-world healthcare settings presents significant challenges [24]. Key concerns include data privacy, model transparency, algorithmic bias, regulatory compliance, and the risk of overreliance on AI in high-stakes clinical decisions. For instance, the use of underrepresented data in training can lead to biased outcomes, amplifying health disparities among patients of color. Moreover, aligning these models with patient-centred values, such as trust, interpretability, and inclusivity, remains an ongoing area of research. As such, understanding the current landscape of patient-centric e-Health and the role HFMs may play in shaping its future is both timely and necessary.

App stores, especially the Google Play Store, have shown great potential as a valuable resource for analyzing trends, identifying popular categories, and understanding the competitive landscape in the digital health space [17, 18]. When developing a new mobile application, it is crucial to understand the surrounding ecosystem, such as the app store environment. Mining app store data provides invaluable insights into widely adopted features and emerging services, enabling companies to identify proven functionalities that can be seamlessly integrated into existing applications. Market entry decisions should be informed by comparative analyses of competing applications, both in terms of functionality and the maturity of those features. This process aligns with the growing emphasis on data-driven approaches in software evolution and requirements engineering, where feature reuse plays a key role in accelerating development and improving software reliability [19, 20]. The use of mobile app stores also taps into the concept of multi-sided markets, where diverse stakeholders, including app developers, end-users, and service providers, can collaborate to offer innovative solutions. This dynamic allows companies to leverage services and features from a wide array of developers, reducing costs and enhancing the overall value proposition of their apps [5, 21, 7]. In the context of digital health, such strategies can significantly improve the functionality and user appeal of apps, positioning them as competitive and feature-rich solutions within the rapidly evolving e-Health ecosystem [125, 126, 127].

In this study, we aim to understand the current landscape of patient-centric e-Health and the role of FMs within this context by examining the e-Health solutions available on the Google Play Store (**Objective 2**). We address the following research questions:

RQ1: What is the current status quo of FM integration in e-Health applications?

To address this question, we identified 116 applications on the Google Play Store that focus on patient support. We systematically mined the data for each application and extracted the functionalities using state-of-the-art methods. By analyzing the features offered by these apps, we assessed the extent to which AI and FM have been integrated into their systems.

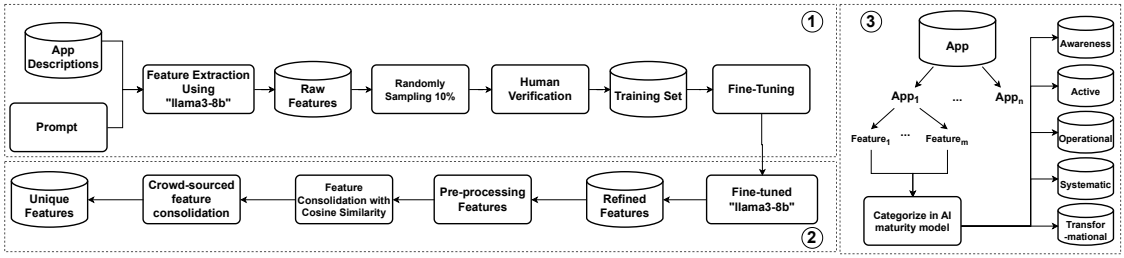

FIGURE 3.1: Overview of the methodology. In part 1, we prepared a dataset to fine-tune the LLaMA model. In part 2, we used the fine-tuned model to extract features from each application, followed by the consolidation of similar features. In part 3, we categorized each application into one of the AI maturity levels based on the Gartner model.

RQ2: Given the growing popularity of FMs, what is the current state of AI maturity in e-Health applications?

To address this question, we utilize the AI maturity model introduced by Gartner Group [68], a widely adopted framework in both research and practice for assessing the level of AI maturity within organizations. We categorized the applications in our dataset according to this model and examined the impact of AI maturity on the popularity of the applications (**Objective 3**).

We review related work in Section 3.2, describe our methodology in Section 3.3, present results in Section 3.4, and discuss findings in Section 3.5. Section 3.6 outlines validity threats, and Section 3.7 concludes the paper.

3.2 Related Studies

AI has seen widespread adoption in healthcare, particularly in diagnostics, treatment planning, and operational efficiency. Secinaro et al. [128] reviewed 288 multidisciplinary studies, identifying key trends in health service management, diagnostics, and predictive medicine. Similarly, Al Kuwaiti et al. [129] emphasized AI's transformative impact, especially during COVID-19, across virtual care, drug discovery, and administration, highlighting ethical, technical, and governance challenges. Despite focusing on the provider side of the applications, AI's role in patient-centric technologies remains underexplored. This is notable given patients' growing use of digital tools for self-monitoring, education, and early diagnosis.

AI maturity models have primarily been developed to assess organizational readiness or the technical robustness of AI systems. These models typically evaluate dimensions such as explainability, adaptability, and data handling capabilities and have been extensively studied in academia [69, 68, 70, 130, 70, 131]. However, their application to consumer-facing healthcare tools is limited. Our work addresses this gap by mapping observable features of healthcare applications to an AI maturity spectrum, offering a novel lens for evaluating their technical evolution and potential risks.

Prior work has used NLP, heuristics, and crowdsourcing to classify healthcare apps by functionality or compliance [132, 133, 134, 135]. While LLMs show promise in Requirements Engineering tasks such as feature extraction, testing, etc., and can approximate human judgment [98, 99, 103, 110], challenges like hallucination and lack of transparency persist. We address this by integrating fine-tuned LLMs with crowdsourced validation for more robust AI feature extraction.

3.3 Methodology

To achieve the objectives of this study, we first extracted e-Health-related applications using `google-play-scraper` [136] library. The applications were identified using the following keywords: "Patient engagement", "Healthcare community", "Chronic disease management", "Health management", "Peer support health", "Patient support network", "Health social network", "Digital health engagement", and "Health community app". For each application, we extracted the following metadata: `appId`, `title`, `score`, `genre`, `price`, `free`, `description`, `developer`, and `installs`.

To identify key functionalities of the apps (**RQ1**), we leveraged LLMs for feature extraction using app descriptions by modifying the pipeline for the feature extraction in Section 2.4. Given the growing success of LLMs in RE tasks and their ability to approximate human evaluation [98, 99, 103, 110], we selected the `llama3.1-8b` model after

manual comparison across candidates. The model was prompted to extract raw features, and a manual review of 146 randomly selected features revealed a 6.16% inaccuracy rate. These were corrected to improve alignment with app descriptions. To enhance accuracy and reduce hallucinations, we fine-tuned the model using the corrected samples and re-ran it on the dataset. Since similar features can appear under different phrasings, we normalized and lemmatized the extracted features, converted them into vector embeddings using `bert-base-nli-mean-tokens`[119], and applied cosine similarity (threshold = 0.9[121]) via `scikit-learn` [120] to cluster and consolidate duplicates. Final refinements were made with human validation through crowd-sourcing. The overview of the feature extraction methodology is depicted in Figure 3.1.

Using these extracted features and discussions with experts in the e-Health application development field, we manually classified each application into one of the levels of the AI Maturity Model (**RQ2**). This framework is widely used in the industry to measure the maturity of companies and their products in terms of AI capabilities [137] (more details in Section 3.3.1). Finally, we performed a comprehensive analysis of the data to assess the impact of FMs on the e-Health landscape, providing insights into how AI influences the development and adoption of e-Health solutions. The replication package used in this study is provided in our GitHub repository [138].

3.3.1 AI Maturity Model

With the growing adoption of AI and FMs, organizations increasingly use AI maturity models (AIMMs) to guide and assess their integration efforts [69]. Among available frameworks, the Gartner AI maturity model [68, 70] is widely referenced in both academia and industry [69, 130, 70, 131]. We adopt this model to evaluate the level of AI integration in patient-centric applications. It defines five distinct levels of maturity:

- *Awareness*: AI-related discussions occur informally and lack strategic direction, with no pilot initiatives underway.
- *Active*: Organizations begin exploring AI through pilot projects or proofs of concept, and preliminary conversations around standardization start to emerge.

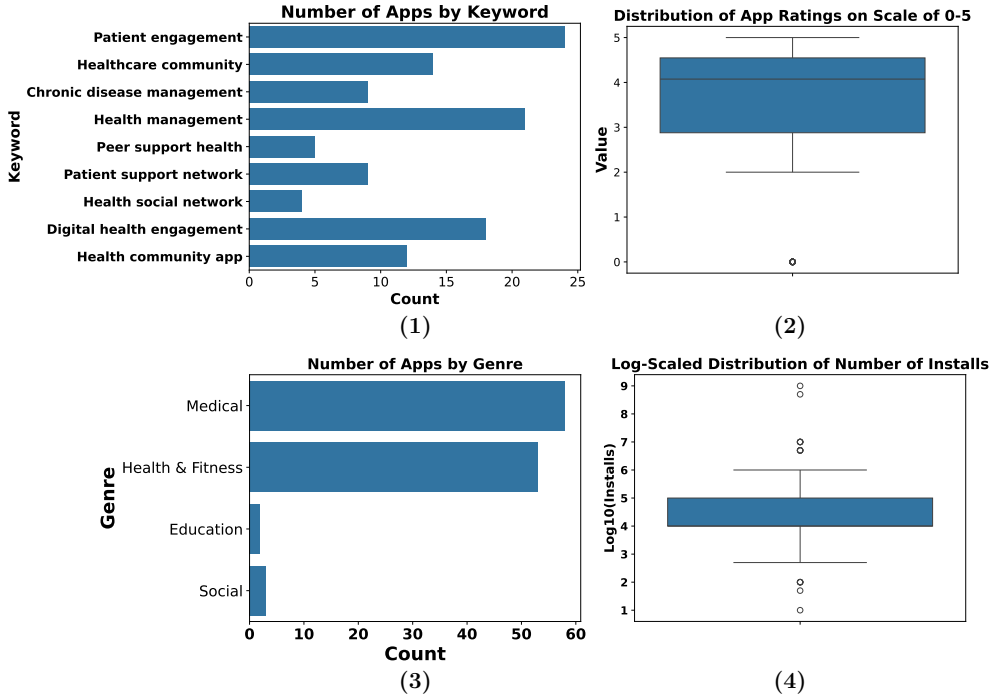

FIGURE 3.2: Analysis patient-centric e-Health apps includes: (1) keyword distribution, (2) ratings distribution, (3) genre distribution, (4) log-scale install count distribution.

- *Operational*: At least one AI initiative is deployed in a production setting, supported by dedicated budgets, executive sponsorship, and access to internal AI expertise and best practices.

- *Systematic*: AI is embedded in the design of new products and services, and employees across departments incorporate AI into processes and applications.

- *Transformational*: Although not explicitly defined in detail, this stage implies a comprehensive and strategic integration of AI that fundamentally reshapes organizational operations, driving sustained innovation and competitive advantage.

Gartner reports [68] highlight that most organizations are still in the early stages of AI maturity, with barriers such as identifying high-impact use cases, privacy concerns, integration challenges, and unrealistic timelines.

3.4 Preliminary Results

We identified 186 applications using the keyword-based approach described in Section 3.3. However, upon manual inspection, we found several applications unrelated to healthcare. Broad terminologies such as "community" and "network" led to the inclusion of general-purpose social media platforms, such as Facebook and Reddit. To ensure the relevance of our dataset, we manually excluded these unrelated entries, resulting in a refined set of 116 patient-centric applications that offer various forms of support to patients. Figure 3.2-1 illustrates the distribution of collected applications based on the keywords used during extraction.

Applications on the Google Play Store can be either free or paid. In our dataset, only one application required payment to install; the rest were free to download, although many included in-app purchases or required referral codes for access to specific features. App ratings and install counts are widely used as proxies for popularity [139, 140, 141]. The average rating across the applications was 3.32, on a scale from 0 to 5, with the distribution shown in Figure 3.2-2.

Given the wide variance in install counts, we applied a logarithmic transformation to normalize the data. After this transformation, the average log-scaled install count was 4.53, corresponding to an average of 33,884 installs per application (Figure 3.2-4). Moreover, the majority of applications in our dataset are categorized under the "Medical" and "Health & Fitness" genres in the Google Play Store, with fewer classified as "Education" and "Social" (Figure 3.2-3).

3.4.1 Functionalities in e-Health Applications (RQ1)

In total, we extracted 942 unique features across 116 applications in our dataset using the methodology outlined in Section 3.3. After consolidating these features to remove duplicates and variations in phrasing, we found that the average number of features per application was 11.12. This suggests that, on average, each application offers a moderate range of functionalities tailored to patient support and care.

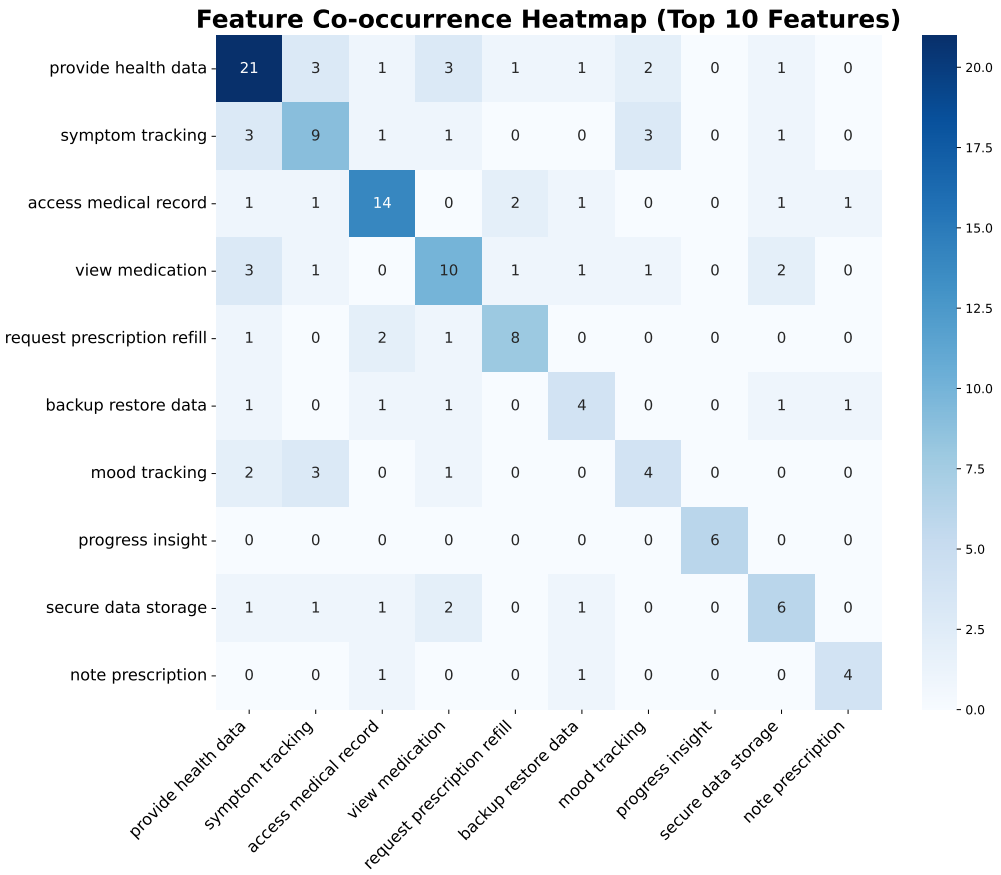

FIGURE 3.3: Top 10 feature co-occurrence (diagonal lines show feature frequency).

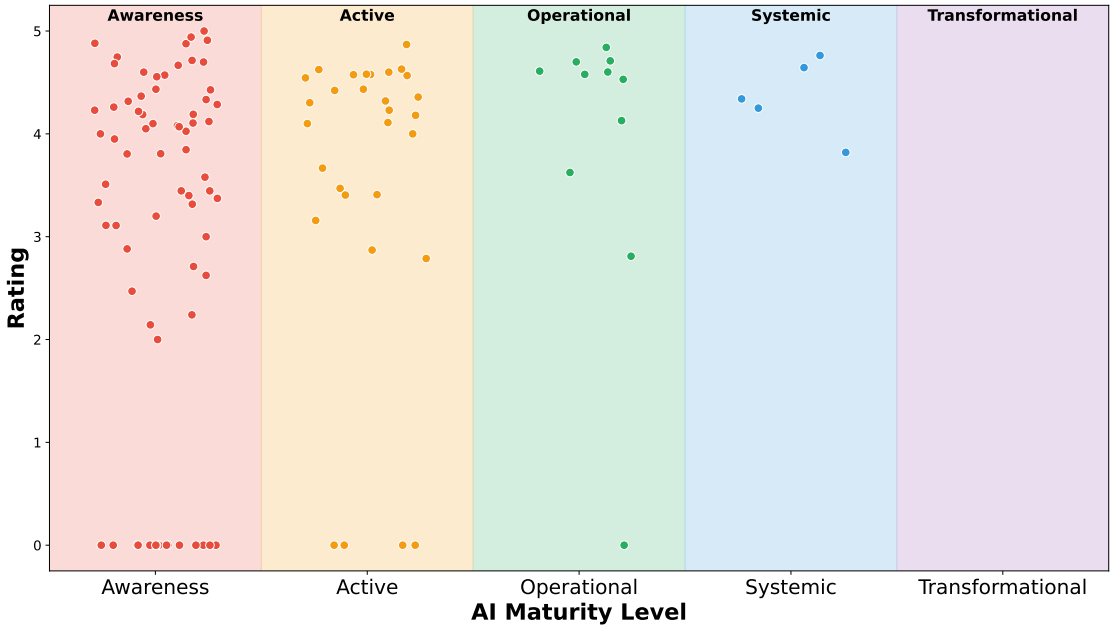

FIGURE 3.4: Categorization of patient-centric e-Health applications by AI maturity level and user rating distribution.

Figure 3.3 presents a heatmap of the top 10 features and their co-occurrence with other features. The diagonal elements of the heatmap represent the frequency with which the apps in our dataset offer each feature. Among the most popular features, we identified “Provide Health Data" (offered 21 times), “Access Medical Record" (14), “View Medication" (10), “Symptom Tracking" (9), and “Schedule Upcoming Appointment" (9). These features stand out as essential components of patient-centric e-Health solutions.

Additionally, our co-occurrence analysis revealed interesting patterns of feature interactions. For instance, “Provide Health Data" showed a weak co-occurrence with both “Symptom Tracking" and “View Medication", suggesting that while the apps commonly offer these features, they are not often combined in the same application. Similarly, “Symptom Tracking" exhibited weak co-occurrence with “Mood Tracking", indicating that while both are related to monitoring patient well-being, they are not frequently integrated in the same app. These findings provide valuable insights into the feature landscape of patient-centric e-Health applications and their design patterns.

3.4.2 Extent of AI Maturity in Digital Health landscape (RQ2)

We manually categorized each application using the extracted features in our dataset based on the levels defined in the AI maturity model (Section 3.3.1), which reflects the extent to which artificial intelligence is integrated into an application's core functionalities. Among the 116 applications analyzed, 70 were classified in the "Awareness" stage, indicating minimal or superficial use of AI, such as basic rule-based systems or static information delivery. This was followed by 30 applications in the "Active" stage, where AI functionalities are present but limited in scope, and 11 in the "Operational" stage, representing more consistent and interactive AI features. Only five applications reached the "Systematic" stage, indicating a deeper and more integrated use of AI across multiple aspects of the application. Notably, no applications were found in the "Transformational" stage, which would indicate a paradigm shift enabled by AI, such as personalized treatment recommendations or autonomous health decision support.

To examine the potential relationship between AI maturity and user perception, we plotted the distribution of applications across AI maturity levels alongside their average user ratings (Figure 3.4). Although a higher AI maturity level does not necessarily guarantee higher user satisfaction, the data indicate a positive trend: Applications in higher AI maturity generally receive better average ratings than those in lower tiers. Additionally, we calculated the correlation between AI maturity levels and app popularity metrics, including user ratings and number of downloads (Figure 3.5). Our analysis revealed a weak but positive correlation (0.30) with both metrics, suggesting that applications may experience positive gains in user engagement and perceived value as AI maturity increases.

3.5 Discussion

Our analysis of 116 patient-centric e-Health apps provides key insights into AI integration and maturity. We will discuss our findings in the following section.

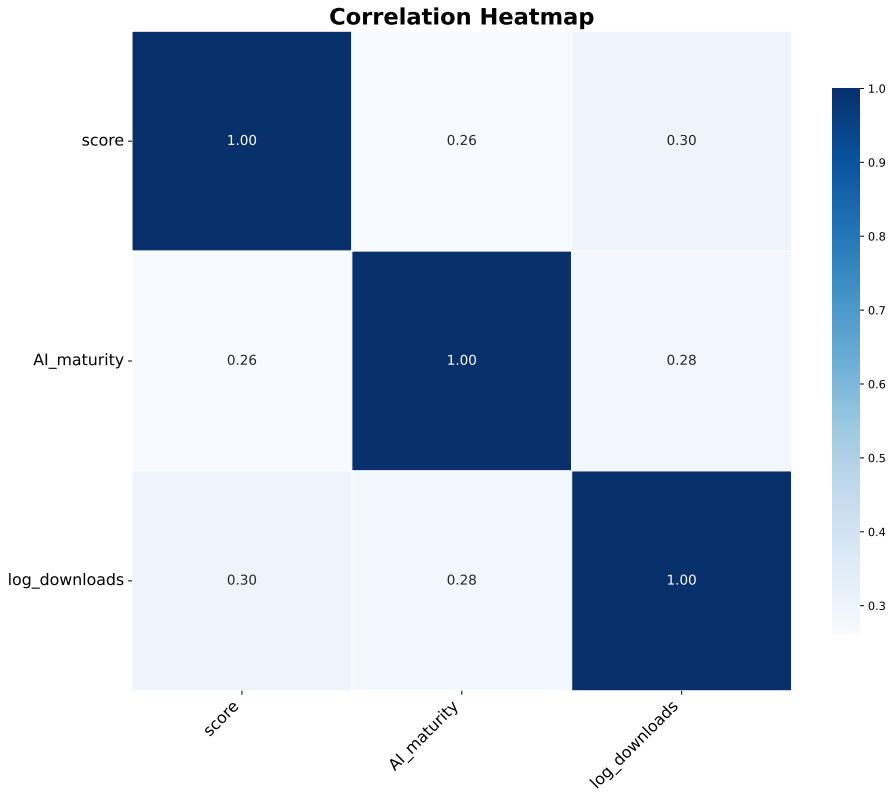

FIGURE 3.5: Correlation matrix between AI maturity, installs, and ratings.

3.5.1 Integration of AI & FMs in e-Health Applications (RQ1)

The feature extraction process revealed that while AI functionalities are present in many patient support apps, they are often limited in scope and depth. Most applications rely on basic automation, chatbots, or rule-based personalization features, suggesting that AI integration is still predominantly at the surface level. Advanced functionalities leveraging deep learning or FMs, such as predictive analytics, intelligent recommendations, or context-aware feedback, remain relatively rare. This gap indicates that the full potential of FMs in e-Health is yet to be realized, particularly in patient-facing solutions.

Additionally, many commonly recurring features (e.g., providing health data, symptom tracking, accessing medical records) do not require high-level AI capabilities but indicate strong UX practices and user-centric design. This suggests that, at present, the value proposition of many e-Health apps lies more in usability and accessibility than in sophisticated AI-driven functionality.

3.5.2 AI Maturity and Its Impact (RQ2)

The classification of applications using the AI Maturity Model revealed that many apps fall within the *Awareness* or *Active* stages. Only a small fraction have reached the *Operational* or *Systematic* levels, with none qualifying as *Transformational*. This finding supports broader observations in the literature [68, 69] that healthcare organizations often struggle to scale AI initiatives beyond isolated use cases.

One important implication is that although the e-Health ecosystem is rich in experimentation, it lacks systemic integration of robust, scalable AI capabilities aligned with clinical workflows. This fragmented maturity also limits the ability of HFMs to deliver on their promise in patient-centric contexts, as their deployment often requires a sophisticated digital infrastructure and trust-enabling mechanisms, such as explainability and compliance frameworks.

Moreover, we observed a positive correlation between higher AI maturity levels and indicators of popularity (e.g., higher ratings or install counts), hinting at a market preference for more intelligent, responsive, and context-aware applications. However,

causality cannot be conclusively established, and further studies are required to examine whether AI maturity directly contributes to user retention and app success or if both are consequences of more capable development teams, better funding, or any other factors..

3.5.3 Implications for Industry and Research

From an industry perspective, our findings highlight an opportunity for developers and healthcare providers to more strategically adopt AI capabilities, especially those enabled by FMs. Rather than focusing solely on technological novelty, developers should prioritize functionalities that demonstrably improve patient engagement, care coordination, and long-term health outcomes. As demonstrated in this study, reusing proven functionalities via app store mining can accelerate the development of robust, AI-enhanced features without reinventing the wheel.

For researchers, this study underlines the need to bridge the gap between FM innovation and real-world implementation in patient-facing tools. Current research in HFMs tends to prioritize model performance on benchmark datasets; however, practical deployment in e-Health settings requires a concurrent focus on usability, safety, interpretability, and alignment with regulatory and ethical frameworks. Future work may explore more nuanced maturity models tailored to healthcare applications, incorporating clinical validation, user trust, and interoperability dimensions.

3.6 Threats to Validity

This section discusses threats to the validity of our study. Despite careful execution, our findings may be influenced by the following:

Construct Validity: Construct validity concerns whether our study accurately captures the concept of AI maturity and functional integration. To ensure the validity of our study, we built on established frameworks and iteratively refined our schema through team discussions. However, classifying AI capabilities and maturity levels inevitably involves subjective judgment, which may have introduced bias.

Internal Validity: Our results are based on observable features extracted from application descriptions during data collection. However, some applications may possess hidden or backend AI functionalities that are not publicly documented. Moreover, we employed LLMs to extract features that introduce the risk of hallucination. To mitigate these concerns, we developed a domain-informed set of keywords to identify patient-centric applications in consultation with experts in the field. Feature extraction was performed using a fine-tuned model, and we manually verified the validity of the extracted features through crowdsourced evaluation.

External Validity: External validity pertains to the generalizability of our findings. Our study focused on a curated sample of 116 patient-centric e-Health applications from a major marketplace, which may not represent other health technologies. However, our dataset's diverse functionality and geographic coverage clearly represent current market practices.

Conclusion Validity: Conclusion validity addresses the reliability and robustness of our interpretations. We refrain from making causal claims while we report descriptive statistics and comparative analyses. Our categorization process, though systematic, is based on qualitative judgments and may be subject to coder bias. To mitigate this, we used team-based coding, maintained detailed decision logs, and reached a consensus on ambiguous cases.

3.7 Conclusion

This Chapter offers a first-of-its-kind analysis of AI maturity and functional integration across 116 patient-centric e-Health applications. Our findings reveal their adoption remains limited and often superficial despite growing interest in AI, particularly Foundation Models. Most applications operate at lower levels of AI maturity, with few leveraging advanced capabilities such as personalized insights or conversational support. Nevertheless, we observe early signals of a shift toward more context-aware, patient-centred AI. As the ecosystem evolves, developers, researchers, and regulators must collaboratively ensure that future integrations are transparent, safe, and aligned

with clinical and user needs. This work lays the groundwork for understanding the emerging role of FMs in e-Health and calls for continued empirical monitoring of their real-world adoption.

Chapter 4

Chapter Four: Conclusion and Future Work

This thesis explores two distinct, yet methodologically aligned, case studies, following the guidelines proposed by Wohlin et al. for designing experiments in the field of software engineering [142]: the GitHub Marketplace as a SECO and the landscape of patient-centric e-Health applications. While each chapter addresses different contexts, open-source automation tools versus clinical digital health, the underlying goals were similar: to analyze functional evolution, understand actor behavior (developers or providers), and investigate the emergence of complex and interconnected functionalities. This work contributes novel insights into how functionalities migrate, interconnect, and mature in dynamic ecosystems by applying empirical methods, feature extraction techniques, and graph-based modeling.

4.1 GitHub Marketplace SECO

In our investigation of the GitHub Marketplace, we analyzed 6,983 Actions and uncovered structural patterns in how automation functionalities are developed and reused. The dataset included 3,869 providers and over 20,000 extracted features. Our analysis

revealed that Actions tend to evolve from independent functionalities to more interconnected, functionally rich tools. Notably, early contributors play a central role in shaping workflows, with their Actions forming the backbone for later developments. The high incidence of feature migration, especially Weak Migration, suggests that developers often adapt and recombine features across contexts, reinforcing the modular and collaborative nature of the SECO. These dynamics point toward an emergent functional synergy among Actions and set the stage for future innovations such as Super Actions.

4.2 Functional Maturity in e-Health Applications

Our second study focused on 116 patient-centric e-Health applications, evaluating their level of functional maturity and depth, with particular emphasis on AI-related features. Despite the increasing prominence of FMs, our findings indicate that most applications still rely on shallow implementations of AI-related features, offering limited personalization, adaptiveness, or contextual intelligence. Only a few systems demonstrated meaningful functional integration, such as conversational agents or dynamic feedback. However, we detected early signs of progress, particularly in applications aiming to become more context-aware and patient-centric. These findings underscore the gap between technological potential and practical implementation in e-Health and highlight the need for regulatory, clinical, and technical collaboration to ensure ethical and effective AI adoption.

4.3 Future Work

While this study offers valuable insights into two distinct types of software ecosystems, the GitHub Marketplace and patient-centric e-Health applications from the Google Play Store, several areas remain open for further investigation. In the context of the GitHub Marketplace, future work could involve understanding and quantifying the feature overlap and intersection among GitHub Actions through traceability techniques. Another venue for future work in this context could be the semantic analysis of feature

content, focusing on specific types of features, such as performance enhancements or integration functionalities, and their influence on ecosystem dynamics. This could help clarify the role of individual features in driving the evolution of Actions, as well as their impact on user adoption and tool innovation. Moreover, a deeper investigation into the concept of Super Actions, comprehensive tools that consolidate multiple functionalities, could provide valuable insights into whether they offer strategic advantages over more modular approaches that use independent Actions to build customized workflows. Future studies could also explore longitudinal changes in provider behavior and feature evolution, examining whether the decline of independent Actions represents a saturation point or a shift toward a more collaborative ecosystem. Additionally, integrating user feedback and real-world usage data, such as adoption rates and developer engagement metrics, could yield a more nuanced understanding of how these evolutionary patterns manifest in practice.

For the e-Health applications, future research should focus on bridging the gap between AI maturity and its practical deployment in real-world healthcare settings. Further investigation is needed to understand how FMs can be better integrated into patient-facing applications, focusing on their usability, transparency, and alignment with clinical workflows. Exploring more sophisticated maturity models tailored to healthcare, incorporating elements like trust, explainability, and regulatory compliance, could be valuable. Additionally, future work could investigate the long-term impact of AI and FM integration on patient outcomes and care coordination, aiming to create a more holistic view of AI's role in transforming e-Health services. Lastly, expanding the analysis of AI-driven functionalities across a broader range of healthcare domains and considering factors like data privacy and model transparency would contribute to a more comprehensive understanding of AI's potential in healthcare.

Bibliography

[1] Geir K Hanssen. "A longitudinal case study of an emerging software ecosystem: Implications for practice and theory". In: *Journal of Systems and Software* 85.7 (2012), pp. 1455–1466.

[2] Rodrigo Santos et al. *Software engineering for systems-of-systems and software ecosystems*. 2024.

[3] Jan Bosch. "From software product lines to software ecosystems." In: *SPLC*. Vol. 9. 2009, pp. 111–119.

[4] Konstantinos Manikas. "Revisiting software ecosystems research: A longitudinal literature study". In: *Journal of Systems and Software* 117 (2016), pp. 84–103.

[5] SK Golam Saroar et al. "GitHub marketplace for automation and innovation in software production". In: *Information and Software Technology* (2024), p. 107522.

[6] Maleknaz Nayebi, Bram Adams, and Guenther Ruhe. "Release Practices for Mobile Apps–What do Users and Developers Think?" In: *2016 ieee 23rd international conference on software analysis, evolution, and reengineering (saner)*. Vol. 1. IEEE. 2016, pp. 552–562.

[7] Maleknaz Nayebi, Homayoon Farahi, and Guenther Ruhe. "Which version should be released to app store?" In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2017, pp. 324–333.

[8] Konstantinos Manikas and Klaus Marius Hansen. "Software ecosystems–A systematic literature review". In: *Journal of Systems and Software* 86.5 (2013), pp. 1294–1306.

[9] Slinger Jansen, Anthony Finkelstein, and Sjaak Brinkkemper. "A sense of community: A research agenda for software ecosystems". In: *2009 31st International Conference on Software Engineering-Companion Volume*. IEEE. 2009, pp. 187–190.

[10] Olavo Barbosa12 and Carina Alves. "A systematic mapping study on software ecosystems". In: (2011).

[11] James F Moore. "Predators and prey: a new ecology of competition". In: *Harvard business review* 71.3 (1993), p. 7586.

[12] Marco Iansiti and Roy Levien. "Strategy as ecology." In: *Harvard business review* 82.3 (2004), p. 6878.

[13] Sk Golam Saroar and Maleknaz Nayebi. "Developers' Perception of GitHub Actions: A Survey Analysis". In: *arXiv preprint arXiv:2303.04084* (2023).

[14] GitHub. *Creating actions Publishing actions in GitHub Marketplace*. Accessed on February 19, GitHub. n.d. URL: https://docs.github.com/en/actions/creatingactions/publishingactionsingithubmarketplace.

[15] Hemank Lamba et al. "Heard it through the Gitvine: an empirical study of tool diffusion across the npm ecosystem". In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 505–517.

[16] Laura Dabbish et al. "Leveraging transparency". In: *IEEE software* 30.1 (2012), pp. 37–43.

[17] Wenlong Xu, Yin Liu, et al. "mHealthApps: a repository and database of mobile health apps". In: *JMIR mHealth and uHealth* 3.1 (2015), e4026.

[18] Mark Erik Larsen, Jennifer Nicholas, and Helen Christensen. "Quantifying app store dynamics: longitudinal tracking of mental health apps". In: *JMIR mHealth and uHealth* 4.3 (2016), e6020.

[19] Rafael Capilla et al. "Opportunities for software reuse in an uncertain world: From past to emerging trends". In: *Journal of software: Evolution and process* 31.8 (2019), e2217.

[20] Hafedh Mili et al. “Discovering reusable functional features in legacy object-oriented systems”. In: *IEEE Transactions on Software Engineering* 49.7 (2023), pp. 3827–3856.

[21] Afnan A Al-Subaihin et al. “App store effects on software engineering practices”. In: *IEEE Transactions on Software Engineering* 47.2 (2019), pp. 300–319.

[22] Rishi Bommasani et al. “On the opportunities and risks of foundation models”. In: *arXiv preprint arXiv:2108.07258* (2021).

[23] Michael Wornow et al. “The shaky foundations of large language models and foundation models for electronic health records”. In: *npj digital medicine* 6.1 (2023), p. 135.

[24] Yuting He et al. “Foundation model for advancing healthcare: challenges, opportunities and future directions”. In: *IEEE Reviews in Biomedical Engineering* (2024).

[25] Noman Ahmed Shaheer and Sali Li. “The CAGE around cyberspace? How digital innovations internationalize in a virtual world”. In: *Journal of Business Venturing* 35.1 (2020), p. 105892.

[26] Jianfeng Guo et al. “Measurement framework for assessing disruptive innovations”. In: *Technological Forecasting and Social Change* 139 (2019), pp. 250–265.

[27] Bharat Rao and Bertha Jimenez. “A comparative analysis of digital innovation ecosystems”. In: *2011 Proceedings of PICMET'11: Technology management in the energy smart world (PICMET)*. IEEE. 2011, pp. 1–12.

[28] Federica Sarro et al. “Feature lifecycles as they spread, migrate, remain, and die in app stores”. In: *2015 IEEE 23rd International requirements engineering conference (RE)*. IEEE. 2015, p. 7685.

[29] Tom Mens et al. “Studying evolving software ecosystems based on ecological models”. In: *Evolving software systems*. Springer, 2013, pp. 297–326.

[30] Umme Ayman Koana et al. “Examining ownership models in software teams: A systematic literature review and a replication study”. In: *Empirical Software Engineering* 29.6 (2024), p. 155.

[31] Faiz Ahmed, Suprakash Datta, and Maleknaz Nayebi. “Negative Results of Image Processing for Identifying Duplicate Questions on Stack Overflow”. In: *arXiv preprint arXiv:2407.05523* (2024).

[32] Umme Ayman Koana et al. “Ownership in the hands of accountability at brightsquid: A case study and a developer survey”. In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 2008–2019.

[33] Maleknaz Nayebi. “Eye of the mind: Image processing for social coding”. In: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 2020, pp. 49–52.

[34] Yalda Hashemi, Maleknaz Nayebi, and Giuliano Antoniol. “Documentation of machine learning software”. In: *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2020, pp. 666–667.

[35] Maleknaz Nayebi et al. “ESSMArT way to manage customer requests”. In: *Empirical Software Engineering* 24 (2019), pp. 3755–3789.

[36] Maleknaz Nayebi, Guenther Ruhe, and Thomas Zimmermann. “Mining treatment-outcome constructs from sequential software engineering data”. In: *IEEE Transactions on Software Engineering* 47.2 (2019), pp. 393–411.

[37] Maleknaz Nayebi. “Analytical Release Management for Mobile Apps”. In: (2018).

[38] Maleknaz Nayebi. “Data driven requirements engineering: Implications for the community”. In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE. 2018, pp. 439–441.

[39] SK Golam Saroar et al. “GitHub Marketplace: Driving Automation and Fostering Innovation in Software Development”. In: *2025 IEEE 32nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. Journal First. 2025.

[40] Federica Pepe et al. “How do Papers Make into Machine Learning Frameworks: A Preliminary Study on TensorFlow”. In: *33rd IEEE/ACM International Conference on Program Comprehension (ICPC 2025)*. 2025.

[41] Faiz Ahmed et al. “Inferring Questions from Programming Screenshots”. In: *22nd International Conference on Mining Software Repositories (MSR)*. 2025.

[42] Maleknaz Nayebi et al. “Recommending and release planning of user-driven functionality deletion for mobile apps”. In: *Requirements Engineering* 29.4 (2024), pp. 459–480.

[43] Maleknaz Nayebi et al. “User driven functionality deletion for mobile apps”. In: *2023 IEEE 31st International Requirements Engineering Conference (RE)*. IEEE. 2023, pp. 6–16.

[44] Maleknaz Nayebi, Henry Cho, and Guenther Ruhe. “App store mining is not enough for app improvement”. In: *Empirical Software Engineering* 23 (2018), pp. 2764–2794.

[45] Maleknaz Nayebi et al. “Crowdsourced exploration of mobile app features: A case study of the fort mcmurray wildfire”. In: *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*. IEEE. 2017, pp. 57–66.

[46] Maleknaz Nayebi and Guenther Ruhe. “Analytical product release planning”. In: *The art and science of analyzing software data*. Elsevier, 2015, pp. 555–589.

[47] Maleknaz Nayebi et al. “A longitudinal study of identifying and paying down architecture debt”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2019, pp. 171–180.

[48] Maleknaz Nayebi et al. “App store mining is not enough”. In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE. 2017, pp. 152–154.

[49] Maleknaz Nayebi and Guenther Ruhe. "Asymmetric release planning: Compromising satisfaction against dissatisfaction". In: *IEEE Transactions on Software Engineering* 45.9 (2018), pp. 839–857.

[50] Walid Maalej, Maleknaz Nayebi, and Guenther Ruhe. "Data-driven requirements engineering-an update". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE. 2019, pp. 289–290.

[51] Maleknaz Nayebi et al. "Anatomy of functionality deletion: an exploratory study on mobile apps". In: *Proceedings of the 15th International Conference on Mining Software Repositories*. 2018, pp. 243–253.

[52] Maleknaz Nayebi and Guenther Ruhe. "Optimized functionality for super mobile apps". In: *RE*. IEEE. 2017, p. 388393.

[53] Guenther Ruhe, Maleknaz Nayebi, and Christof Ebert. "The vision: Requirements engineering in society". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE. 2017, pp. 478–479.

[54] Maleknaz Nayebi et al. "Analytics for Software Project Management–Where are We and Where do We Go?" In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*. IEEE. 2015, pp. 18–21.

[55] Maleknaz Nayebi and Guenther Ruhe. "An open innovation approach in support of product release decisions". In: *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*. 2014, pp. 64–71.

[56] Maleknaz Nayebi, Homayoon Farrahi, and Guenther Ruhe. "Analysis of marketed versus not-marketed mobile app releases". In: *Proceedings of the 4th International Workshop on Release Engineering*. 2016, pp. 1–4.

[57] Maleknaz Nayebi et al. "Hybrid labels are the new measure!" In: *IEEE Software* 35.1 (2017), pp. 54–57.

[58] Günther Ruhe and Maleknaz Nayebi. "What counts is decisions, not numbers—toward an analytics design sheet". In: *Perspectives on Data Science for Software Engineering*. Elsevier, 2016, pp. 111–114.

[59] Maleknaz Nayebi and Guenther Ruhe. "Analytical open innovation for value-optimized service portfolio planning". In: *Software Business. Towards Continuous Value Delivery: 5th International Conference, ICSOB 2014, Paphos, Cyprus, June 16-18, 2014. Proceedings 5*. Springer. 2014, pp. 273–288.

[60] Maleknaz Nayebi et al. "More insight from being more focused: analysis of clustered market apps". In: *Proceedings of the International Workshop on App Market Analytics*. 2016, pp. 30–36.

[61] Shaikh Jeeshan Kabeer et al. "Predicting the vector impact of change-an industrial case study at brightsquid". In: *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE. 2017, pp. 131–140.

[62] Maleknaz Nayebi and Guenther Ruhe. *Trade-off service portfolio planning–a case study on mining the android app market*. Tech. rep. PeerJ PrePrints, 2015.

[63] Xuchen Tan et al. "ImageR: Enhancing Bug Report Clarity by Screenshots". In: *arXiv preprint arXiv:2505.01925* (2025).

[64] Sharuka Promodya Thirimanne et al. "One Documentation Does Not Fit All: Case Study of TensorFlow Documentation". In: *arXiv preprint arXiv:2505.01939* (2025).

[65] N Maleknaz and R Guenther. "Trade-off Service Portfolio Planning–A Case Study on MiningtheAndroid App Market". In: *PeerJPrePrints | https://dx. doi. org/10.7287/peerj. preprints. 1354v1 | CC-BY* 4 ().

[66] Sk Golam Saroar, Waseefa Ahmed, and Maleknaz Nayebi. "GitHub Marketplace for Practitioners and Researchers to Date: A Systematic Analysis of the Knowledge Mobilization Gap in Open Source Software Automation". In: *arXiv preprint arXiv:2208.00332* (2022).

[67] Luis E Hestres. "App neutrality: Apple's app store and freedom of expression online". In: *International Journal of Communication* 7 (2013), p. 15.

[68] Gartner, Inc. *The CIO's Guide to Artificial Intelligence*. https://www.gartner.com/smarterwithgartner/the-cios-guide-to-artificial-intelligences. Accessed: 2025-04-07. 2018. URL: https://www.gartner.com/smarterwithgartner/the-cios-guide-to-artificial-intelligence.

[69] Sulaiman Alsheibani, Yen Cheung, and Chris H Messom. "Towards An Artificial Intelligence Maturity Model: From Science Fiction To Business Facts." In: *PACIS*. 2019, p. 46.

[70] Ulrich Lichtenthaler. "Five maturity levels of managing AI: From isolated ignorance to integrated intelligence". In: *Journal of Innovation Management* 8.1 (2020), pp. 39–50.

[71] NetworkX Developers. *NetworkX Documentation*. Accessed on February 19. NetworkX Developers. n.d. URL: https://networkx.org/.

[72] Rodrigo Santos et al. *Software engineering for systems-of-systems and software ecosystems*. 2023.

[73] Pablo Antonino et al. "Report on the 12th ACM/IEEE International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems-SESoS@ ICSE 2024". In: *ACM SIGSOFT Software Engineering Notes* 49.3 (2024), pp. 53–56.

[74] William Martin et al. "A survey of app store analysis for software engineering". In: *IEEE transactions on software engineering* 43.9 (2016), pp. 817–847.

[75] Jean-Charles Rochet and Jean Tirole. "Platform competition in two-sided markets". In: *Journal of the European Economic Association* 1.4 (2003), pp. 990–1029.

[76] David S Evans. "The antitrust economics of two-sided markets". In: *Yale Journal on Regulation* 20 (2003), p. 325.

[77] C Sutcu. "Two-Sided Markets: Apple's Digital Application Platform". In: *International Journal of Economics and Financial Issues* 9.1 (2019), pp. 1–7.

[78] Félix Cuadrado and Juan C Dueñas. “Mobile application stores: success factors, existing approaches, and future developments”. In: *IEEE Communications Magazine* 50.11 (2012), pp. 160–167.

[79] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. “On the rise and fall of CI services in GitHub”. In: *SANER*. IEEE. 2022, p. 662672.

[80] JV Joshua et al. “Software ecosystem: Features, benefits and challenges”. In: *International Journal of Advanced Computer Science and Applications* 4.8 (2013).

[81] Xuemei Xie and Hongwei Wang. “How can open innovation ecosystem modes push product innovation forward? An fsQCA analysis”. In: *Journal of Business Research* 108 (2020), pp. 29–41.

[82] Matthijs Den Besten et al. “Collaboration and innovation dynamics in software ecosystems: A technology management research perspective”. In: *IEEE Transactions on Engineering Management* 68.5 (2020), pp. 1532–1537.

[83] Eric Yu and Stephanie Deng. “Understanding software ecosystems: A strategic modeling approach”. In: *Iwseco-2011 software ecosystems 2011. proceedings of the third international workshop on software ecosystems. brussels, belgium*. 2011, pp. 65–76.

[84] Piers RJ Campbell and Faheem Ahmed. “A three-dimensional view of software ecosystems”. In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. 2010, pp. 81–84.

[85] Mahsa Hasani Sadi and Eric Yu. “Analyzing the evolution of software development: From creative chaos to software ecosystems”. In: *2014 IEEE eighth international conference on research challenges in information science (RCIS)*. IEEE. 2014, pp. 1–11.

[86] Paulo Malcher et al. “Investigating Open Innovation Practices to Support Requirements Management in Software Ecosystems”. In: *International Conference on Software Business*. Springer Nature Switzerland Cham. 2023, pp. 35–50.

[87] Tingting Chen et al. “Let’s Supercharge the Workflows: An Empirical Study of GitHub Actions”. In: *QRSC*. IEEE. 2021, p. 0110.

[88] Pablo ValenzuelaToledo and Alexandre Bergel. “Evolution of github action workflows”. In: *SANER*. IEEE. 2022, p. 123127.

[89] Alexandre Decan et al. “On the use of github actions in software development repositories”. In: *ICSME*. IEEE. 2022, p. 235245.

[90] Timothy Kinsman et al. “How do software developers use github actions to automate their workflows?” In: *MSR*. IEEE. 2021, p. 420431.

[91] Ali Khatami, Cédric Willekens, and Andy Zaidman. “Catching Smells in the Act: A GitHub Actions Workflow Investigation”. In: *24th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM). IEEE*. 2024.

[92] Pablo Valenzuela-Toledo et al. “The Hidden Costs of Automation: An Empirical Study on GitHub Actions Workflow Maintenance”. In: *arXiv preprint arXiv:2409.02366* (2024).

[93] Antonio Mastropaolo et al. “Toward Automatically Completing GitHub Workflows”. In: *ICSE*. 2024, p. 112.

[94] Chaminda Chandrasekara et al. “Introduction to github actions”. In: *Hands-on GitHub actions: implement CI/CD with GitHub action workflows for your applications* (2021), pp. 1–8.

[95] Mairieli Wessel et al. “Github actions: the impact on the pull request process”. In: *Empirical Software Engineering* 28.6 (2023), p. 131.

[96] Alexandre Decan, Tom Mens, and Hassan Onsori Delicheh. “On the outdatedness of workflows in the GitHub Actions ecosystem”. In: *Journal of Systems and Software* 206 (2023), p. 111827.

[97] Mark Harman, Yue Jia, and Yuanyuan Zhang. “App store mining and analysis: MSR for app stores”. In: *MSR*. IEEE. 2012, p. 108111.

[98] Andreas Vogelsang. “From Specifications to Prompts: On the Future of Generative Large Language Models in Requirements Engineering”. In: *IEEE Software* 41.5 (2024), p. 913.

[99] Quim Motger et al. “Tfrex: A transformerbased feature extraction method from mobile app reviews”. In: *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2024, p. 227238.

[100] Christof Ebert and Panos Louridas. “Generative AI for software practitioners”. In: *IEEE Software* 40.4 (2023), pp. 30–38.

[101] Sandeep Singh Sengar et al. “Generative artificial intelligence: a systematic review and applications”. In: *Multimedia Tools and Applications* (2024), pp. 1–40.

[102] Daniel Russo. “Navigating the complexity of generative ai adoption in software engineering”. In: *ACM Transactions on Software Engineering and Methodology* 33.5 (2024), pp. 1–50.

[103] Lianmin Zheng et al. “Judging llm-as-a-judge with mt-bench and chatbot arena”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 46595–46623.

[104] Christoph Seidl and Uwe Aßmann. “Towards modeling and analyzing variability in evolving software ecosystems”. In: *Proceedings of the 7th International Workshop on Variability Modelling of Software-intensive Systems*. 2013, pp. 1–8.

[105] Jieun Kim et al. “Mobile application service networks: Apple’s App Store”. In: *Service Business* 8 (2014), p. 127.

[106] Python Software Foundation. *Beautiful Soup Documentation*. Accessed on February 19. Python Software Foundation. n.d. URL: https://pypi.org/project/beautifulsoup4/.

[107] SecretActionDeveloper. *Extension Decisions in Open Source Software Ecosystem*. Accessed: 2024-12-31. 2024. URL: https://github.com/SecretActionDeveloper/Extension-Decisions-in-Open-Source-Software-Ecosystem/tree/main.

[108] Zhiyuan Zeng et al. “Evaluating large language models at evaluating instruction following”. In: *arXiv preprint arXiv:2310.07641* (2023).

[109] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. "Judgelm: Fine-tuned large language models are scalable judges". In: *arXiv preprint arXiv:2310.17631* (2023).

[110] Aparna Elangovan et al. "Considers-the-human evaluation framework: Rethinking human evaluation for generative large language models". In: *arXiv preprint arXiv:2405.18638* (2024).

[111] OpenAI. *Best Practices for Prompt Engineering with the OpenAI API*. Accessed: 20240814. 2024. URL: https://help.openai.com/en/articles/6654000bestpracticesforpromptengin

[112] Mo Wang et al. "Unleashing ChatGPT's power: A case study on optimizing information retrieval in flipped classrooms via prompt engineering". In: *IEEE Transactions on Learning Technologies* (2023).

[113] Rui Zhong et al. "Leveraging large language model to generate a novel metaheuristic algorithm with CRISPE framework". In: *Cluster Computing* (2024), p. 135.

[114] Groq. *Groq*. Accessed: 20240905. 2024. URL: https://groq.com/.

[115] Xinyi Hou et al. "Large language models for software engineering: A systematic literature review". In: *ACM Transactions on Software Engineering and Methodology* 33.8 (2024), pp. 1–79.

[116] Mona Nashaat and James Miller. "Towards efficient fine-tuning of language models with organizational data for automated software review". In: *IEEE Transactions on Software Engineering* (2024).

[117] Jiho Shin et al. "Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks". In: *arXiv preprint arXiv:2310.10508* (2023).

[118] Potsawee Manakul, Adian Liusie, and Mark JF Gales. "Selfcheckgpt: Zero-resource blackbox hallucination detection for generative large language models". In: *arXiv preprint arXiv:2303.08896* (2023).

[119] Hugging Face. *bertbasenlimeantokens*. https://huggingface.co/sentencetransformers/bertbasenlimeantokens. Accessed: 20240905. 2024.

[120] Scikitlearn. *cosine similarity*. https://scikitlearn.org. Accessed: 20240905. 2024.

[121] Muhammad Abbas et al. "On the relationship between similar requirements and similar software: A case study in the railway domain". In: *Requirements Engineering* 28.1 (2023), p. 2347.

[122] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. *NetworkX*. Accessed: 2024-08-28. 2024. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.degree_centrality.html.

[123] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. *NetworkX: Connected Components*. Accessed: 2025-02-28. 2024. URL: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.components.connected_components.html.

[124] Sebastián Pizard, Diego Vallespir, and Barbara Kitchenham. "A longitudinal case study on the effects of an evidence-based software engineering training". In: *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*. 2022, pp. 1–13.

[125] Lapo Filistrucchi. "Market definition in multi-sided markets". In: *Rethinking antitrust tools for multi-sided platforms* 37 (2018).

[126] Bruno Jullien. "Competition in multi-sided markets: Divide and conquer". In: *American Economic Journal: Microeconomics* 3.4 (2011), pp. 186–219.

[127] Guofu Tan and Junjie Zhou. "The effects of competition and entry in multi-sided markets". In: *The Review of Economic Studies* 88.2 (2021), pp. 1002–1030.

[128] Silvana Secinaro et al. "The role of artificial intelligence in healthcare: a structured literature review". In: *BMC medical informatics and decision making* 21 (2021), pp. 1–23.

[129] Ahmed Al Kuwaiti et al. "A review of the role of artificial intelligence in healthcare". In: *Journal of personalized medicine* 13.6 (2023), p. 951.

[130] Daniel E O'Leary. "The impact of Gartner's maturity curve, adoption curve, strategic technologies on information systems research, with applications to artificial intelligence, ERP, BPM, and RFID". In: *Journal of Emerging Technologies in Accounting* 6.1 (2009), pp. 45–66.

[131] Sergey Yablonsky. "AI-driven platform enterprise maturity: from human led to machine governed". In: *Kybernetes* 50.10 (2021), pp. 2753–2789.

[132] Elias Hossain et al. "Natural language processing in electronic health records in relation to healthcare decision-making: a systematic review". In: *Computers in biology and medicine* 155 (2023), p. 106649.

[133] Claudio Crema et al. "Natural language processing in clinical neuroscience and psychiatry: A review". In: *Frontiers in Psychiatry* 13 (2022), p. 946387.

[134] Ian James Bruce Young, Saturnino Luz, and Nazir Lone. "A systematic review of natural language processing for classification tasks in the field of incident reporting and adverse event analysis". In: *International journal of medical informatics* 132 (2019), p. 103971.

[135] Kerri Wazny. "Applications of crowdsourcing in health: an overview". In: *Journal of global health* 8.1 (2018), p. 010502.

[136] JoMingyu. *google-play-scraper*. Accessed: 2025-04-07. 2024. URL: https://pypi.org/project/google-play-scraper/.

[137] Leinar Ramos, Anthony Mullen, and Pieter den Hamer. "The CIO's Guide to Building an AI Roadmap That Drives Value". In: *Gartner* (2025). Accessed: 2025-04-07. URL: https://www.gartner.com/smarterwithgartner/the-cios-guide-to-artificial-intelligence.

[138] SecretActionDeveloper. *The Impact of Foundational Models on Patient-Centric eHealth Systems*. https://github.com/SecretActionDeveloper/The-Impact-of-Foundational-Models-on-Patient-Centric-eHealth-Systems. Accessed: 2025-04-14. 2025.

[139] Anthony Finkelstein et al. “Investigating the relationship between price, rating, and popularity in the Blackberry World App Store”. In: *Information and Software Technology* 87 (2017), pp. 119–139.

[140] Henrik Sällberg, Shujun Wang, and Emil Numminen. “The combinatory role of online ratings and reviews in mobile app downloads: an empirical investigation of gaming and productivity apps from their initial app store launch”. In: *Journal of Marketing Analytics* 11.3 (2023), pp. 426–442.

[141] Yuan Tian et al. “What are the characteristics of high-rated apps? a case study on free android applications”. In: *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE. 2015, pp. 301–310.

[142] Claes Wohlin et al. *Experimentation in software engineering*. Springer Science & Business Media, 2012.