Bughunting on a Budget: Exploring Quality Assurance Practices and Tools for Indie Game Developers

by

Jaehyung Cho

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

The games industry is growing worldwide, eclipsing the global film industry as a premier entertainment solution. The development of a commercial game is a complex, lengthy, and costly process, and increasing amounts of resources are required to build a successful game as technology improves. To this end, quality assurance (QA) is critical for producing high-quality games that are fun and reasonably defect-free. Existing work in QA for games describes game development methodologies and testing approaches, goals, and automation, but do not address the disparate contexts of independent (indie) and non-indie game development. Indie developers have limited access to funding and resources compared to their non-indie counterparts, which makes achieving a similar quality of QA difficult. Yet, indie games make up the lion's share of newly released games each year. Therefore, we want to empower indie developers by maximizing their QA opportunities within their resource constraints.

In this study, we survey 19 game developers who have experience with commercially released games to learn about their QA experiences and perspectives based on 22 of their released game projects. We compare responses about indie and non-indie projects regarding test performance, planning, goals, automation, tools, results, and resources to find key differences in practices, goals, and needs between them. We present our findings on the major differences in game testing and automation between indies and non-indies, including indies having less clear goals and plans for testing, performing tests on a conditional basis over a regular

testing schedule, and having subjective test results which are open to interpretation. We also discuss issues of culture and management that respondents raise in their open-ended answers. Based on our findings, we present recommendations for each area of testing to improve quality assurance practices for resource-constrained game developers.

Preface

We plan to publish the details and findings of our survey in the future.

The survey in this study, "Quality Assurance Practices, Goals, and Needs in Game Development", was approved through the University of Alberta Research Ethics Board for human studies, ID# Pro00108719, 25 Jan 2021.

Acknowledgements

I want to express my gratitude to those without whom this research and thesis would not be possible:

First, to my supervisor, Dr. Karim Ali, who first sparked my interest in academic research and continued to mentor me throughout my undergraduate and Master's. His encouragement, patience, and kindness during many changes and challenges were a lifeline during my studies, and I would not be where I am today without the time and opportunities he has given me over the years.

I would also like to thank colleagues at the Maple Lab for their help and support throughout my studies, and my colleagues at Caldera Interactive for their flexibility and cooperation. Special thanks to Dr. Sarah Nadi who helped review my methodology and patiently answered my questions. I am also grateful to all of the survey respondents for the time they took to offer their perspective and input.

This research would not have been possible without funding from the Canada Graduate Scholarship from the Natural Sciences and Engineering Research Council of Canada; the Accelerate Fellowship from Mitacs in cooperation with Synopsys Canada; the Graduate Fellowship from the University of Alberta; and the Post-Graduate Training on Scholarship program from the Canadian Armed Forces.

Last but not least, my sincerest gratitude to my friends and family for believing in me, helping with the survey, and cheering me on to the finish line. I could not have dreamed of doing this without the support of my partner and my two incredible kids.

Contents

1 Introduction **1**

2 Background **5**
2.1 Game Development 5
2.2 Game Testing 7
2.3 Literature Review 9
2.3.1 Game Development Methodologies 9
2.3.2 General Software Development vs. Game Development . 10
2.3.3 Testing in Games 11
2.3.4 Bugs in Games 11
2.3.5 Automated Testing in Games 12
2.4 Summary 14

3 Game Testing Survey **15**
3.1 Study Design 15
3.1.1 Respondent Information 15
3.1.2 Project-Based Responses 17
3.2 Pilot Survey 20
3.2.1 Pilot Survey Recruitment 21
3.2.2 Pilot Survey Participants 21
3.2.3 Pilot Survey Evaluation 22
3.2.4 Intermediate Discussion 24
3.3 Survey Updates from Pilot 24
3.3.1 Changes to Respondent Information 24
3.3.2 Changes to Project-Based Responses 25
3.3.3 General Testing 26
3.3.4 Participant Recruitment 26
3.4 Methodology 27
3.4.1 Data Verification and Grouping 27
3.4.2 Quantitative Analysis 28
3.4.3 Qualitative Analysis 29
3.5 Summary 34

4 Survey Results **36**
4.1 Respondent Information 36
4.2 Project-based Responses 38
4.3 Data Organization 39
4.4 Quantitative Differences between Indie and Non-Indie Responses 42
4.4.1 Comparing Indie and Non-Indie Response Medians 42
4.4.2 Comparing Indie and Non-Indie Response Distributions . 43
4.4.3 “Not Applicable” and “Unknown” Responses 45
4.5 Qualitative Analysis 46

4.5.1 Testing Approach . . . 46
4.5.2 Testing Goals . . . 49
4.5.3 Test Plans . . . 51
4.5.4 Test Types . . . 52
4.5.5 Testing Tools . . . 54
4.5.6 Testing Resources . . . 56
4.5.7 Test Timing . . . 58
4.5.8 Test Automation . . . 60
4.5.9 Test Results . . . 62
4.5.10 Overall Sentiment . . . 64
4.6 General Testing . . . 65
4.6.1 Helpful for Testing . . . 65
4.6.2 Pain Points for Testing . . . 66
4.6.3 Wants for Testing . . . 67
4.7 Summary . . . 68

5 Discussion of Results **70**
5.1 Goals and Planning . . . 70
5.2 Testing Procedures and Timing . . . 71
5.3 Resources . . . 72
5.4 Results and Outputs . . . 74
5.5 Automation . . . 75
5.6 Tools . . . 76
5.7 Culture . . . 78
5.8 Research Questions . . . 79
5.8.1 What are the most significant differences in game testing between indie and non-indie developers? . . . 79
5.8.2 How much automated testing do indie developers use compared to non-indie developers, and what are the biggest pain points for automation? . . . 79
5.8.3 Where should indie developers focus their testing efforts and allocate their testing resources to maximize their return on investment? . . . 80
5.9 Limitations and Threats to Validity . . . 81
5.10 Summary . . . 82

6 Conclusion **84**

References **87**

Appendix A Pilot Survey Questions **96**
A.1 Respondent Information . . . 96
A.1.1 Information and Consent . . . 96
A.1.2 Eligibility . . . 96
A.1.3 Demographics . . . 96
A.2 Project-Based Responses . . . 97

Appendix B Game Testing Survey Questions **103**
B.1 Respondent Information . . . 103
B.1.1 Information and Consent . . . 103
B.1.2 Eligibility . . . 103
B.1.3 Demographics . . . 103
B.2 Project-Based Responses . . . 104
B.3 General Testing . . . 111

Appendix C Respondents' Demographic Information 112

Appendix D Project Information 113

Appendix E Distributions of Likert-like responses per prompt 114

Appendix F Boxplots of Likert-like responses per prompt 122

List of Tables

2.1 Guidelines for classifying project types. 6

3.1 Demographics of pilot survey respondents. 21
3.2 Project information for pilot survey responses. 21

4.1 Project information for indie games. 39
4.2 Project information for non-indie games. 39
4.3 Differences in Likert-like response medians between indies and non-indies in decreasing order. A positive delta indicates stronger agreement from non-indies than indies, and vice-versa. 42
4.4 Prompts with significant differences ($\alpha = 0.025$) in responses between indies and non-indies. A higher U-statistic or CLES indicates stronger agreement from indies than non-indies, and vice-versa. Sample size is 11 for all samples except medians marked with an asterisk* to indicate a sample size of 10 or a dagger† for a sample size of 9. 44
4.5 Six projects with the highest opt-out responses in descending number of opt-outs. 45

C.1 Respondents' demographic information 112

D.1 Project-based information for game submissions 113

List of Figures

4.1 Boxplot of respondents' experience in game development. 37
4.2 Roles that respondents held over their careers. 37
4.3 Game submissions by type. 38
4.4 Game engines used on submitted projects. 38
4.5 Budget ranges for submitted games. 38
4.6 Team size ranges for submitted games. 38
4.7 Boxplots of responses for the top ten differences in medians. The text of the corresponding prompts are shown in Table 4.3. 41

E.1 Distributions of Likert-like responses per prompt 115

F.1 Boxplots of Likert-like responses per prompt 122

Glossary

Common Language Effect Size (CLES)

Entertainment Software Rating Board (ESRB)

feature-driven development (FDD)

Games-as-a-Service (GaaS)

Monte Carlo tree search (MCTS)

Quality Assurance (QA)

Reflexive Thematic Analysis (RTA)

test-driven development (TDD)

Unified Modeling Language (UML)

User Experience (UX)

Chapter 1

Introduction

The games industry is constantly growing worldwide, being worth over $150B USD in 2020 and $180B USD in 2021 [19], a sharp rise compared to its total worth of under $8B USD in 2000 and under $20B USD in 2009 [42]. Games are a premier entertainment industry which out-earned the global film industry and North American sports industries combined in 2020 [102], fuelled in part by the COVID-19 pandemic environment which saw a need for accessible and engaging entertainment at home and the ability to connect with others virtually. Games also serve important societal and educational functions—74% of parents in the United States play video games with their children at least once a week, and 80% of Americans believe that games are educational [1]. Despite some arguments to the contrary [27], games are also largely considered to be an art form, with selected video games having been curated and displayed at the Smithsonian American Art Museum [88] and the Museum of Modern Art [78]. In short, video games are no longer the niche that they once were; they are a global industry with wide reach and social impact.

The design and development of commercial games is a complex, lengthy, and costly process that continues to increase in complexity, development duration, and cost with new developments in the state of the art [84]. A former Sony CEO estimated the rise in cost for a blockbuster PlayStation console game from $100M per title for the PlayStation 4 to $200M per title for the PlayStation 5 [73]. Given these rising costs, game studios are under pressure to maximize their return on investment and minimize unnecessary or unforeseen costs. Quality Assurance (QA) and testing is one aspect of development that game studios can leverage to

increase efficiency and decrease potential downstream costs. Bugs and defects in software are costlier to fix the later they are discovered in the development cycle; fixing a bug found in testing can be 15 times more costly than if it had been found while it was in design [21], [67]. It is then in the developers' best interests for bugs to be detected and corrected as early in the process as possible. While blockbuster—so-called "AAA"—games and studios have millions in funding, specialized teams focused on specific aspects of the game, and dedicated QA teams for testing with software and hardware infrastructure provided for them, smaller independent (i.e., "indie") studios have major resource constraints that limit their QA abilities. For example, they may be a small team without the finances to hire QA testers, lack the variety of hardware required for full coverage of configuration testing, or not have the proficiency or time to employ testing methodologies or tools which could increase efficiency and effectiveness.

Although they have fewer resources, indie games are an important pillar of the gaming industry. Due to differing definitions of what comprises an indie game or studio, there are no universally agreed-upon statistics on indie vs. non-indie development. By way of illustration, on Steam [82], the leading digital PC game storefront, indie games account for over 95% of all titles listed, under 40% of units sold, and 28% of revenues [29]. However, despite the size and importance of the games industry and the value of early detection and correction of game defects, there remains a lack of academic research on game development and testing as a discipline of software engineering. Additionally, QA in the games industry continues to be the subject of debate and stigma, with QA testers being looked down upon as unskilled labor, receiving low pay, and having low job security [46] [72].

In this thesis, we seek to examine the current practices, goals, and needs of commercial game developers to identify how we can improve quality assurance practices and tools for resource-constrained game developers. In particular, we are trying to answer the following research questions:

RQ1 : What are the most significant differences in game testing between indie and non-indie developers?

RQ2 : How much automated or scripted testing do indie developers used compared to non-indie developers, and what are the biggest pain points for automation?

RQ3 : Where should indie developers focus their testing efforts and allocate their testing resources to maximize their return on investment?

The major contribution of this thesis is a review of the current practices, goals, and needs of commercial indie game developers, which we achieve through an anonymous online survey of 19 game developers who detailed their previous experiences with QA on 22 commercially-released game projects. To our knowledge, this is the first study specifically examining the differences in testing practices, priorities, and tools between indie and non-indie game development. This survey will be the focus of the thesis.

To answer our research questions, we conduct a review of the current literature relating to testing in games as well as differences between indie and non-indie game development, which we discuss in Chapter 2. We also explain key terms and concepts related to the thesis in this chapter to provide further background on the wide range of practices in commercial game development. Based on our findings in the literature, we create an online survey to solicit developer retrospectives of commercially released games focusing on seven primary areas of interest: test performance, test planning, testing goals, test automation, testing tools, test results, and testing resources. We compare the feedback we receive between indie and non-indie projects to address our research questions.

We first conduct an initial pilot survey to validate the survey and the rationale behind its structure and questions, which we discuss in Chapter **??**. We then review our takeaways from the pilot and the feedback that we receive from pilot participants. The full text of the pilot survey is available in Appendix A. In Chapter 3, we discuss the full game testing survey that we deployed online. We first lay out the survey design, focusing on what we changed based on the pilot survey, then discuss our methodology for processing and analyzing the collected data. We conduct both quantitative and qualitative analyses on the responses to

gain as much insight into our research questions as we can. The text of the game testing survey is available in Appendix B.

We present our analyses of the results in Chapter 4, starting with the demographics of the respondents and an overview of the game projects received. We detail our findings from the quantitative and qualitative analyses as well as key takeaways from the data. We discuss the results in more detail, along with their implications and potential shortcomings in Chapter 5. Finally, Chapter 6 contains a summary of the thesis and potential avenues of future research. Artifacts for this study consist of the survey questions, anonymized answers, and analysis results, which are available in the appendix and online [18].

Chapter 2

Background

This chapter presents background information on commercial game development and testing. We introduce terms and concepts important to the industry and repeated throughout this thesis, and provide an overview of the prior work informing our study.

2.1 Game Development

Video games may be categorized in many different ways. They may be broken down by the hardware used to play them, such as PC and consoles; their graphics and artistic representation, such as pixel-art or 3D; the game mechanics and conventions they contain, broadly described as game genres, such as platformers or first-person shooters; the number and connection of players, such as single-player or online multiplayer; and many more. Some of these categories are well-defined and widely known, whereas others are loosely defined and have conflicting interpretations. In this section, we clarify our usage of some common terms.

One major category used in games is the type or scope of a game, which are often referred to as "AAA", "AA", or "indie". AAA games represent major blockbuster games—usually, well-known titles from high-profile game studios with massive budgets—whereas AA games are big games from stable studios that do not have the reach, marketing, and development power for a AAA game. Indie games are generally from smaller studios that do not have access to as many resources, and lack publishing and marketing support. There are no clear industry definitions

Table 2.1: Guidelines for classifying project types.

Classification	Team Size	Budget	Development Time
Indie	1–49	< $1M USD	1–2 Years
AA	50–99	$1–10M USD	2–4 Years
AAA	>= 100	> $10M USD	>= 4 Years

on what makes a game indie or not [100], and even broadly accepted conventions have obvious exceptions. The two primary conventions of indie games are financial independence and creative independence. For financial independence, people generally consider a game to be indie if it is developed without major funding or publisher support, with the developers shouldering the bulk of development costs or crowdfunding the project. For creative independence, people generally consider a game to be indie if it is developed without external oversight or control with respect to the game content and direction [58].

However, there are famous indie games that are not financially or creatively independent. For example, the game *Journey* from the humorously-named studio thatgamecompany is classified as "Indie" on Steam [36] and Wikipedia [37] although it had funding and publishing support from Sony Computer Entertainment. A recent indie publishing contract made public [99] shows how prevalent and exploitative such contracts are—including a clause where, should there be disagreement over the game's direction, the publisher can unilaterally bring in a third-party developer to create their vision of the game at the indie developer's expense. Additionally, there are categorizations outside of these three scopes, such as "AAA+" games, which emerged in the 2010s to describe games with ongoing monetization, such as massive multiplayer online games or Games-as-a-Service (GaaS) approaches that involve recurring subscriptions. "III" applies to indie games that have budgets and game sizes comparable to AAA games. However, these terms are still less used than indie, AA, and AAA classifications.

For the purposes of our study, we ask respondents to classify games as AAA, AA, or indie according to three factors: the size of the team, the project budget, and the length of development. Table 4.5 lists the ranges we gave as guidance for classification, where a game should meet at least two of the criteria in the row

to be so classified. For example, a game made by 20 developers with a $5M USD budget in two years would be considered Indie. If a game did not fit two criteria, respondents were asked to use their best judgment in classification. We chose these levels based on the levels of different games and studios popularly thought to be indie, AA, or AAA, with the understanding that these are not strict definitions but rough guidelines [83]. Since our goal is to focus on games and studios with resource constraints, it is suitable for our purposes that the indie classification necessarily means the fewest resources in terms of team members, funding, and time.

While AAA and AA games are commercial endeavors with large budgets and marketing efforts, indie games can be commercial or non-commercial. For example, Digital: A Love Story [44] is a non-commercial indie game released for free that gained thousands of fans and critical success [45]. Our study focuses specifically on commercial games to be directly comparable to commercial AAA and AA games, as non-commercial games do not have the same requirements, costs, or considerations, especially for quality assurance.

2.2 Game Testing

The goal of quality assurance for games is to minimize the number and impact of bugs, defects, and other unwanted behaviors or artifacts that make it to the release product. It can take many forms depending on the game, studio, and resources. In this section, we cover a non-exhaustive list of common types of testing in games, all of which have been mentioned in the survey responses.

Many types of testing that are done for games mirror the types of testing done for general software development. For terms that are relatively well-known in the general software development field, such as build verification tests, integration tests, regression tests, smoke tests, stress tests, unit tests, and user experience (UX) tests, we do not provide definitions in this section. Other tests in this section may not be specific to the games industry but may have particular considerations or implications for games.

Alpha tests encompass every feature and their expected behaviors and may be a combination of manual and automated tests. They are usually the largest suite of tests and are conducted periodically, including before major releases. If the game supports multiple platforms, there is a series of alpha tests for each platform.

Balance tests are done to ensure that the game mechanics work well with each other over the course of gameplay and present a game with meaningful player choices and resource management.

Boundary tests refer to testing interactions between systems at boundary values, but in games can also refer to testing the limits of the game space and ensuring that players cannot cross the boundaries meant to confine them to a certain space.

Combat tests are done on games with heavy combat mechanics to ensure that combat interactions function as expected; this can be a combination of many other types of tests applied to the combat portions of the game. This can include testing animations, hitboxes, boundaries, damage, health, balance, and more.

Core and Critical Path tests are like their general software development counterpart in that they test the most important features and functions that the majority of users will use most frequently. For games, this primarily involves testing the main game path and progression, if there is one, including any branching and multiple endings that may be possible. This is related to *Progression tests* which test for game progression regardless of whether it is on the core path.

Compliance and Certification tests are done to ensure that the game meets compliance or certification requirements for a release platform or console. Game stores and consoles have stringent QA requirements and standards that, if not met, could end in the game's rejection. Testing for compliance with ratings boards, such as Entertainment Software Rating Board (ESRB), are also in this category.

Configuration and Compatibility tests involve testing the game with different combinations of software and hardware to ensure compatibility and operation. For games, this includes testing with different processors, graphics cards, memory, peripherals, console versions, controllers, and more.

Experiential tests collect feedback on how the game feels to play to validate player-focused design. These tests have less to do with functionality and mechan-

ics and more to do with player perception and emotion during gameplay.

Greybox tests can have similar meanings to general software development (i.e., testing with partial knowledge of internal systems) but can also mean testing of intended game design and mechanics using quick, low-fidelity placeholder assets. For example, a developer might test a platformer game concept using simple boxes for platforms, obstacles, and characters to prove the concept prior to spending time creating and implementing polished assets.

Playtests involve playing through the game in part or in whole as a player would. They are one of the most important and frequently performed tests in game development. This is where a large amount of exploratory or ad-hoc testing occurs.

Soak tests refer to leaving the game running over long periods of time in different states, which can detect unwanted behaviors, memory leaks, and other rare conditions.

While not a type of test in itself, *Embedded testing* refers to when QA testers are located with the development teams, versus *Non-embedded testing* where QA testers are separate from the rest of the team.

2.3 Literature Review

2.3.1 Game Development Methodologies

We first want to understand the environment in which game developers work, starting with their development methodologies. Some methodologies, such as test-driven development (TDD) specify philosophies and approaches to QA; others indirectly affect testing priorities and plans.

An analysis of 20 game post-mortems by Politowski et al. [61] find that iterative, agile processes are becoming more commonplace in game development, with 55% of respondents using iterative processes, but that these methodologies are not always properly applied or understood. McKenzie et al. [50] interview 10 game studios from New Zealand about their development methodologies and find that all of them use Scrum, half use Kanban, and one-third use feature-driven develop-

ment (FDD), though studios unknowingly overestimate their adherence to Scrum principles in practice. Musil et al. [54] survey 13 studios in Austria and find similar results in studios using agile methodologies, primarily Scrum; they also find that only 30.8% of studios use automated testing, and that while all surveyed studios conduct testing in-house, major studios outsourc QA, particularly to their publishers.

2.3.2 General Software Development vs. Game Development

Since agile approaches are also mainstream in general software development, we compare it to game development to find key differences in goals and methodologies. We want to know whether research on testing in software development could be transferrable to testing in games.

Murphy-Hill et al. [53] conduct qualitative interviews with 14 participants who have experience with both game development and general software development experience to identify major differences, then validate them with a quantitative survey of 364 developers at Microsoft across game and non-game departments. They find that practices used in software engineering are not well-suited for games, primarily because the goal of game development is *fun*, which is highly subjective and difficult to form objective requirements around. They note little reuse of tools and code between games since games need project-specific performance tuning, and low use of automation due to its cost and fragility to frequent changes.

These findings are confirmed by Politowski et al. [63], who conduct a review of 96 academic papers as well as gray literature (i.e., post-mortems, game development conferences, web articles) on QA for games. They find that game testing is focused on the player experience rather than functional accuracy, game requirements are different from general software requirements, code is not often reused between games, and there is not much low-level testing or automation. They also find in the gray literature that developers feel there is insufficient testing overall and that they have difficulty setting up testing tools.

2.3.3 Testing in Games

Understanding that there are fundamental differences in requirements and practices for game testing, we next drill deeper into *what* game developers test and *how* in order to understand the core activities involving game testers.

Politowski et al. [64] analyze 927 problems from 200 post-mortems from games between 1997 to 2019 that they divide into 20 categories. From the testing category, the most mentioned issues are insufficient test coverage and issues with process and testing plans, followed by specific project requirements and a scope too large to properly test. The postmortems mention playtesting but not unit or integration tests or automated testing.

Kasurinen and Smolander [40] conduct semi-structured interviews with 27 game developers from southeast Finland to find out what and how game developers test their products. They apply grounded theory to their interviews and find three main categories for testing: game mechanics, encompassing the game's rules, balance between features, and overall design; technical aspects, such as functionality and stability of the software; and user experience, focusing on the fun of the game, player impressions, and overall satisfaction with the game. The developers they interviewed state that user experience testing is the most important piece, and that they consider themselves to be creatives, not software engineers.

Marklund et al. [48] review 48 papers published between 2006 and 2016 and find game development as a discipline to be chaotic with no standard methods or practices shared between all studios. They find that playtesting is critical at every stage of the development process, but that features being open to change throughout all of development often leads to feature creep.

2.3.4 Bugs in Games

We also examined papers discussing video game bugs to understand the kinds of bugs that game developers regularly encountered, and what kinds of defects are possible in games that may not exist in most general software development.

Lewis et al. [43] present a taxonomy of video game bugs formed from videos of game issues uploaded to YouTube. They present two major categories, temporal

and non-temporal bugs, where a non-temporal bug can be identified as a bug at any point in time but a temporal bug requires past context in order to determine that it is faulty. Examples of non-temporal issues include misbehaving AI agents and lack of required information; examples of temporal issues include invalid positioning over time and unwanted event interruptions.

Truelove et al. [91] expand upon this taxonomy by analyzing 12,122 bugfixes from 30 popular Steam games and collating their frequency. The most frequently recurring bugs they saw are game crashes, graphical bugs, and triggered event bugs; the most severe bugs are game crashes, object persistence, and triggered events. Through a survey of 47 game developers, they identified the major issues with game testing as a lack of certain kinds of testing (automated, integration, cross-platform), inadequate testing, reproducing bugs, and code quality.

2.3.5 Automated Testing in Games

We also want to look specifically at automation for testing in games. Automation is mentioned in most of the papers that we reviewed, but does not appear to be widely used or understood. As one of the major problems with game QA is resource constraints and endless brute-force manual testing, automation is an attractive potential solution for tedious and lengthy tests, especially for teams without the resources to hire dedicated testers.

Politowski et al. [62] perform a literature review of 166 papers on game testing automation from 2004 to 2021, and conduct a survey using the seven most promising automated testing methods from the literature to assess their desirability, viability, and feasibility. While they are not able to draw conclusions from the data, some concerns raised by respondents include the amount of data and time required to train AI agents and the cost of setting up the process from the ground up. Respondents state that they want automated testing tools that are easy to maintain and not specific to the game being tested, but think that the majority of game testing will still be manual in 10 years.

In a comprehensive literature review and classification of automated game testing methods, Albaghajati and Ahmed [3] identify five primary categories of auto-

mated tests: search-based, goal-directed, human-like, scenario-based, and model-based, as well as the primary testing goals for each type. They discuss a number of shortcomings in the field, including a lack of automated methods to verify procedurally generated content, difficulties in avoiding agent bias for certain genres, and game testing examples having limited state space complexity, such as tile-matching games. As their study is current and comprehensive, we review some of their insights below.

Search-based algorithms explore the state space of a game and analyze states according to predetermined criteria. These algorithms include evolutionary algorithms, such as genetic algorithms [15], [30], [68], [85] and graph-based algorithms, such as Monte Carlo tree search (MCTS) [33], [41], [51], [109]. They are best suited for testing exploration, reachability, and survivability. However, finding the best fitness agent for evolutionary algorithms could be time-consuming, and the representations of game state space are limited.

Goal-directed approaches define policies, rewards, and penalties to induce the agent to behave in desirable ways according to predefined objectives. There are reinforcement learning implementations that are used for playtesting, balancing, and level design [8], [55], [60], [75]. Some specialized heuristic solutions are highly performant and effective at identifying faulty states and balance issues [35], [76], [77]. Training reinforcement learning agents is time-consuming and better-suited to particular genres, whereas specialized heuristics approaches are generally subjective and specific to the game being tested.

Human-like approaches are made to mimic human play, used for testing aspects such as emotional response, curiosity, and aggressiveness. Machine learning approaches primarily use player data for agent training and are used to find gameplay flaws, identify game control issues, and train game AI [10], [32], [59], [79], [80], [108]. Limitations of this method include a lack of player data and the space complexity of the game which would affect agent performance. MCTS-based implementations of humanlike play are used for playtesting and balance, but are difficult to generalize to other games [4], [5], [22], [52], [81].

Scenario-based approaches use sequences of actions from recorded human ac-

tions or game simulations to perform tests. Recorded actions include recording and playing back game events in order, or capturing and replaying user inputs to test the game automatically; these solutions are difficult to generalize and fragile to level updates [14], [57]. Simulations use the game's grammar and virtual maps to automatically test defined scenarios [16], [17], [38].

Model-based approaches turn the game into abstract representations in order to verify sequences of events. Some approaches use petri nets to verify errors, check counterexamples, and identify deadlocks, but are dependent on scene-based games and the underlying engine [66], [103]. Other approaches use Unified Modeling Language (UML) to create state machine models and testcases, but require software engineering knowledge and proper UML modeling for games in order to be useful [34], [70].

These studies also note that automated testing in games is recently rising in popularity, with the number of articles published about the subject increasing as of 2017. Despite a slight drop in articles in 2020, we are hopeful that more novel methods of automated testing that are easier to set up and use for resource-constrained projects will become available over time.

2.4 Summary

In this chapter, we introduced key terms and concepts–such as indie, AA, and AAA game development–and explained selected types of tests important to games. We provided an overview of relevant literature concerning QA for games, including studies of game development methodologies, differences between game development and general software development, how testing is performed for games and what is tested, and what kinds of bugs are frequently encountered. We also looked at the state of applicable automated testing methods, their strengths and drawbacks, and recently rising popularity.

Chapter 3

Game Testing Survey

To answer our research questions and gather information on the state of testing for indie quality assurance, we conduct an online survey of commercial game developers. This chapter discusses the survey layout and rationale, how it changed through pilot testing, and the methodology we use to analyze the results.

3.1 Study Design

We use SurveyMonkey [89] to create our survey and deploy it online. It consists of two sections: respondent information and project-based responses.

3.1.1 Respondent Information

This section of the survey is concerned with the respondent's information and consent, eligibility, and demographics.

Information and Consent

Respondents are shown the information and consent page upon visiting the survey. It displays all of the consent and ethics information, including a downloadable PDF version of all of the information shown. We gather explicit consent from the participant in order to continue.

Eligibility

Eligible participants for this survey have (1) at least two years' experience in commercial game development in the past ten years, and (2) worked for at least one

year on a game that was commercially released. We select these criteria to restrict responses to the domain of commercial games, because the dynamics of hobbyist or amateur games have different considerations. Since our goal is to help developers who want to sustain themselves in the profession of game development, our focus is on commercial and released projects.

We specify two years of experience in commercial game development to select participants who have enough time in the industry to potentially observe different teams, projects, project management styles, and leadership styles, and to develop skills in their respective fields to be able to offer thoughtful rationale behind their responses. We choose a minimum of one year of experience on a game that was commercially released to differentiate from the general commercial game development requirement, as working on a commercially released game and understanding release management and post-release support leads to specific pressures and stressors not experienced on non-released commercial games.

Only participants who indicate that they meet both eligibility criteria continue on to the next section; one or more "No" responses results in disqualification.

Demographics

After confirming consent and eligibility, we request some simple demographic data from respondents. We ask them how long they have been active in the games industry, what roles they have previously held, and how many years of indie, AA, and AAA game development experience they have.

These questions are used to confirm eligibility requirements, to gain a sense of how much experience respondents have, and how different experiences may affect respondents' views and priorities. We want to ensure a good mix of experiences to capture a diversity of opinions, as priorities and procedures can differ vastly between studios or even between products at the same studio.

The groups of roles for past experience are chosen for the following considerations:

- *Developer, Programmer, Engineer* ("Dev"): technical roles concerned with implementing the features of the game; usually most involved with the imple-

mentation and technical details

- *Quality Assurance, Tester* ("QA"): dedicated testing roles; usually most involved with testing and test documentation tasks
- *Team Lead, Producer, Project Manager* ("Lead"): supervisory and management roles with input into production planning, prioritization, and feature inclusion; usually have say over what makes it into the final game and what gets cut
- *Game Designer, Level Designer, Gameplay Designer, Product Owner* ("Designer"): design and management roles with input into feature details, intended game aesthetics and gamefeel; usually have say over how features and content should behave and feel
- *Other*

3.1.2 Project-Based Responses

Following the respondent information section, we ask participants to provide responses pertaining to one commercial game project with which they have at least one year of development experience. We require respondents to provide responses for at least one project, but they could optionally provide responses for up to three projects.

The project-based responses begin by collecting project information. There are seven main parts for gathering quality assurance data, organized by topic: test performance, test planning, testing goals, test automation, testing tools, test results, and testing resources. These categories are chosen based on our survey of existing literature, experience in the games industry, and our research questions. Each of these sections contains one open-ended question followed by a series of prompts to be answered on a Likert-like scale. Respondents could optionally provide additional information for the project at the end of the project-based responses in an open-ended text field.

Project Information

For project information, we ask participants to choose any nickname or codename they would like to refer to the project and specify that it does not have to be the name of the game itself for anonymity purposes. This is to make ease of reference easier on survey questions and to be clear which project we are asking about for participants who provide responses for more than one game project. We ask respondents to classify the game as indie, AA, or AAA; what their primary and other roles on the project were; the game engine that was used; the length of time they worked on the project; the size of the project team; approximate budget; publisher support; and length of overall development. These are gathered for statistical purposes and to understand the context of the responses.

Test Performance

In this section, we ask an open-ended text question asking respondents to describe how testing was performed on the project - what steps were taken for testing, who performed the tests, and when/how often testing was performed. We give eight prompts to be answered on a Likert-like scale where we ask respondents to rate their agreement with the given statements about testing procedures. These include questions about the types of tests that were run, whether testing tools were third-party or developed in-house, whether there was a dedicated QA team, and how much agency the team had in the testing process.

Test Planning

Here, we ask respondents about who made the test plans, how, and when in the open-ended text prompt. The rating prompts probe into what department generally set priorities, how well the test plan was known and available to everyone, and whether those plans were generally followed.

Testing Goals

For goals, we want to see how well the processes described by respondents align with their actual goals for testing. We ask what the goals were and how their test-

ing process met those goals or fell short of them. The rating prompts dig deeper into the frequency of testing by various goals, such as systems verification, functional testing, player experience testing, as well as the perceived subjectivity of testing results.

Test Automation

In the automation section, we ask respondents to describe the degree and type of manual and automated testing they did, with a focus on automation. We also ask them to specify whether certain aspects or areas of the game were targeted specifically for automated or manual testing to try to gain a better understanding of where automated testing can be most effective. The additional prompts are about the proportion of automated to manual testing, maintenance of automation, comparative actionability of the results, and importance of each type of testing.

Testing Tools

We are interested in learning what tools were most popularly used, whether there is any correlation between tools being used (for example, if the use of one tool is strongly correlated with the use of another), and the fit of the tools being used. We ask respondents to list what tools they used, whether they were used in a certain order, and what the major benefits and drawbacks were. For rating agreement, we prompt respondents about the usability and relevance of the tools they used, the overlap and interoperability of the tools, and the usability of their outputs.

Test Results

We then focus in on test results and outputs, asking participants about their format, readability, and usability as well as how the results were used. In the rating prompts, we ask about specific use cases for results for different departments and the impact the results could have on the overall design and development of the game.

Testing Resources

Last but not least, we ask respondents about various testing resources and their allocation, including human resources, time, funding, and hardware. We want to know the extent of testing capacity they had on the project compared to how much they felt they needed, and what resources were most scarce for different respondents. We also ask whether testers were embedded into development teams or separate from implementation altogether.

Additional Information

Once the respondent completes all of the sections, they are asked if they want to provide any additional details for the project as an open text response, and whether they want to discuss another eligible game project. If the respondent opt to discuss another game, they are given another set of project-based responses, or prompts PQ9–PQ34.

There are 10 project information questions and 60 quality assurance prompts for a total of 70 responses per project-based response set. Including the 8 respondent information questions, respondents answered 78 questions if they provided information on one project, 148 if they provided information on two projects, or 218 if the provided information on three projects.

3.2 Pilot Survey

Prior to deploying our survey for all respondents, we conduct an initial pilot of our survey to identify any gaps and flaws. Our goal is to ensure that the questions and prompts make sense to respondents, the flow of the survey is logical, the conditional branching works as expected, and the responses are reasonable to analyze and interpret. We use the prefix **PQ** when discussing questions from the pilot survey to indicate its source and distinguish it from questions from the full survey. The full text of the pilot survey can be found in Appendix A.

Table 3.1: Demographics of pilot survey respondents.

Participant	Total Gamedev Experience	Roles Previously Held	Indie Gamedev Experience	AA Gamedev Experience	AAA Gamedev Experience
A	2 years	Quality Assurance/Game Tester Team Lead/Producer/Project Manager	3 years	2 years	0 years
B	3 years	(None of the Above)	2 years	1 year	1 year
C	2 years	Quality Assurance/Game Tester Team Lead/Producer/Project Manager	0 years	2 years	0 years

Table 3.2: Project information for pilot survey responses.

Participant	Project Scope	Primary Project Role	Game Engine	Project Involvement	Project Team Size	Approx. Project Budget	Publisher Support
A	AA Game	Team Lead/Producer/ Project Manager	Unreal 4	<= 1 year	50–99	$1–$10MM	No
B	AAA Game	Other - Narrative Designer	Proprietary	<= 1 year	>= 500	> $10MM	Yes, with Marketing
C	AA Game	Quality Assurance/ Game Tester	Electron	2–3 years	1–49	$1–$10MM	Yes, without Marketing

3.2.1 Pilot Survey Recruitment

We piloted the survey several times ourselves to check for logic, branching flow, and mistakes. We also invited 3 game developers (labeled Participants A through C) from diverse backgrounds to pilot the survey and give us feedback on each question as well as the overall flow and feel of the survey. The diversity of backgrounds includes gender minority (women, trans men), experiences (AAA, AA, Indie), positions (QA, Producer, Writer), and game engines (Unreal Engine, Electron/Web, Proprietary).

The chosen pilot participants are game developers that the author knew from an undergraduate game development certificate program that they took together. We contacted and recruited them through Discord, an online chat app, and all three participants completed one pass of the survey for one project each.

3.2.2 Pilot Survey Participants

The 3 pilot survey participants report between 2–3 years of experience in game development across various scopes and roles. They afford a good coverage of potential survey responses and sufficient context and perspective for evaluating the relevance and effectiveness of the survey logic and prompts as shown in Table 3.1.

Table 3.2 shows that each participant submitted 1 project response, totalling

2 AA and 1 AAA game projects. These projects vary in game engine, team size, budget, and publisher support. Despite not submitting an indie project for the pilot survey, the two respondents who have indie development experience (Participants A and B) provided feedback during the debrief with their indie development experience also in mind. The author's own experience with indie game development was also considered when designing and evaluating the survey.

3.2.3 Pilot Survey Evaluation

During the pilot survey, we were less concerned with the responses themselves and more concerned with the overall progression, coherence, and relevance of the survey. All participants were asked to submit as many project-based responses as practicable, and each participant completed the survey with one project submission. Due to their limited time in the game industry, they did not have a second project which qualified for inclusion. We tested to ensure that the survey software functioned as expected for two and three project-based responses as well.

Following the pilot survey completion, we conducted debriefing interviews with each participant, which lasted about 20 minutes on average. Participants were asked to keep notes as they progressed through the survey with any questions, concerns, or feedback that they had. Following a verbal debrief in person or via voice call, the notes that each participant took were also collected digitally for reference. The feedback collected include:

- Definitions of Indie, AA, AAA should be repeated wherever they are presented as choices (e.g., definitions given for PQ6 but not PQ10)
- If an "Other" option is available, the respondent should be allowed to specify the answer with a short text answer (e.g., PQ5)
- Include definitions for unit, integration, smoke, and regression tests for PQ20
- For Likert-like responses, there should be an "Unknown" option for cases where the prompt is applicable but the respondent does not know the answer

- The list of roles that respondents have held did not account for creative-focused roles such as Artist, Audio, and Narrative disciplines
- Length of development did not seem like an important factor for determining the scope of a game project, as the respondent had experience with shorter AAA games and longer Indie games
- The collection and use of some of the requested information was concerning in terms of being potentially identifying for the individual, such as PQ13 about the game engine used. For projects with in-house engines, this question alone could potentially identify the game or franchise; paired with other information gathered such as the respondent's primary role, this could identify individuals
- The open-ended prompts should indicate that the respondent can include as much or as little information as they see fit

From the information collected on the survey software, Participant A completed the survey in 28 minutes, Participant B in 37 minutes, and Participant C in 28 minutes. The average time to completion was 31 minutes, which was within the expected range of 20–40 minutes given by the authors and lower than the 56 minutes estimated by SurveyMonkey. Our target for an average response with one project was 20–30 minutes.

We analyzed the responses on a participant-by-participant basis for consistency within responses as well as on a prompt-by-prompt basis for consistency between responses. Open-ended text responses ranged between 10–160 words each, and described a diversity of testing processes, goals, and resources between the three projects. On Likert-like responses, we looked for the number of N/A responses entered as an indicator of relevance. Participant A had no N/A responses, whereas Participant B had 29 and Participant C had two. Upon closer examination, the reason for the high number of N/A responses for Participant B was due to their role as a Narrative Designer with minimal involvement in some areas of testing. They were therefore unable to answer all of the questions in the Test Automation,

Testing Tools, and Test Results sections due to the specific and applied nature of testing internally on the narrative team, which was done manually with a narrow range of criteria, such as story development and voiceover bugs.

3.2.4 Intermediate Discussion

All participants have relevant experience with testing in game development across different roles, project types, game engines, and development processes, with appropriate experience and context to offer insight. Participant B has less intimate and technical knowledge of the testing process outside of their domain from their open-ended text responses as well as a high number of N/A responses. This may be due to their primary role as a Narrative Designer which is less involved with the administration and oversight of testing tasks outside of specific domain-related tasks. Another contributing factor may be the massive size of their team with over 500 personnel, increased division of labor, and more compartmentalized information. This changed our expectations in responses for the full survey and led to the inclusion of the "Unknown" response option to account for larger teams and specialized roles with less knowledge or visibility of testing on other aspects of the game.

3.3 Survey Updates from Pilot

The updated survey follows the same general layout as the pilot survey with two major sections: respondent information and project-based responses. Based on the pilot survey, we added a small section at the end for final comments on testing in video games at large with three open-ended text questions. We prefix survey questions with **Q** to differentiate them from the pilot survey's **PQ** prompts.

3.3.1 Changes to Respondent Information

In the demographics portion, we update Q5 to include an "Other" option with a write-in text field for any roles we do not explicitly offer as a choice:

(Q5) *Which of the following roles have you previously held in the games industry?*

Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner; Other (write-in text answer)

Additionally, we add the following questions in the demographics portion to capture additional data:

(Q9) *What is your age range?* Select: Under 18; 19–24; 25–34; 35–44; 45–54; 55–65; 65+

(Q10) *What country do you currently reside in?* Text input answer

3.3.2 Changes to Project-Based Responses

We add the criteria for classifying a project as being Indie, AA, or AAA with recommended ranges of team size, project budget, and project duration to any question dealing with these distinctions instead of only being included in the initial respondent information page. We update the project information question about engine usage from *What engine did you use for the project?* to append *If it was a proprietary engine that you do not want to identify due to privacy reasons, please enter "Proprietary"* for privacy and anonymity purposes.

For each of the seven open-ended text entry responses, the direction *Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying* is added to the end. This is due to some of the brief, one-sentence responses that we observed in the pilot study that we want to avoid in this survey. We also add definitions for unit, integration, smoke, and regression tests to the Test Performance questions.

We add an "Unknown" option for each set of Likert-like prompts. Respondents are able to choose between *Strongly Disagree, Somewhat Disagree, Neither Agree nor Disagree, Somewhat Agree, Strongly Agree, N/A,* or *Unknown.* This option is added to be able to differentiate between items that are not relevant to the project and items that may be relevant but are unknown to the respondent.

3.3.3 General Testing

We add this section to the end of the survey from seeing a lack of information on general testing processes and tools during the pilot, as respondents focused on project-specific experiences without comparing them to the larger state of the practice. Our aim is to gain insight into larger trends in testing for games and to elicit more opinionated responses for how developers want testing for games to change. The three questions we ask are:

(Q88) *What has made testing easier or faster for you in the past?* Text entry response

(Q89) *What are the biggest pain points you have encountered with testing?* Text entry response

(Q90) *What changes would you like to make or see for testing for games?* Text entry response

There are 10 project information questions and 60 quality assurance prompts for a total of 70 responses for project-based response set. Including the 10 respondent information questions and three general testing questions, respondents answered 83 questions if they provided information on one project, 153 if they provided information on two projects, or 223 if they provided information on three projects.

3.3.4 Participant Recruitment

We set out to reach as many game developers with commercial experience as possible to take the survey. Since we had no guaranteed way of verifying an anonymous respondent's credentials, we used non-probability sampling. We advertised the survey on social media platforms such as Twitter, Facebook, LinkedIn, Slack, and Discord, including public game development-focused groups and communities. To catch as many game developers as possible regardless of geographic location or time zone, we made these social media posts periodically and at varying times of day. We additionally asked colleagues and industry contacts to invite any

eligible game developers they knew for snowball sampling. We ensured that the participants of the survey did not overlap with the participants of the pilot survey.

3.4 Methodology

3.4.1 Data Verification and Grouping

Our first step was checking the integrity of the responses we received and using only valid responses for data analysis. Out of 82 people who began the survey, 21 respondents completed and submitted their responses. We discarded the incomplete responses, leaving us with 21 entries, which we examined for completeness and coherence. We looked through the respondent information to verify that their responses made sense—checking for obvious responses that were out of range or answers that contradicted one another. For Likert-like responses, we checked that participants did not appear to have answered at random or without care, such as selecting "N/A" or predictable patterns for all questions.

After this check, we removed two responses due to unsuitability. One respondent appeared to be a troll who wrote *"You lost me at 'he/him', pronouns in bio is a red flag."* for every open text response in response to the author including his pronouns on his social media account. The other respondent did not provide usable information with open text responses such as simply *"me"* for describing the test planning process and only *"Test is performed by other people"* as the answer for five other questions. There were other respondents who were unable to answer certain questions for sections of the survey, such as those who did not have experience with automated testing or those who had no visibility into test planning processes. This was evidenced by a high number of "N/A" or "Unknown" responses in these sections. However, we opted to retain these responses as they were potentially indicative of trends in the industry and their open text responses provided additional insight into their experiences and opinions.

In the end, we had 19 eligible respondents who submitted a total of 22 accepted project-based responses, with one respondent submitting 2 projects and one respondent submitting 3 projects. Each respondent was assigned a respondent code

(*R1–R19*) and each project was assigned a project code (*P1–P22*). After processing, the projects were split into two equal groups: indie and non-indie, each with 11 games; we discuss the data organization in more detail in Chapter 5. We used quantitative analysis for the Likert-like responses and qualitative analysis for the open text responses to analyze these two groups.

3.4.2 Quantitative Analysis

Our goal in quantitative analysis is to determine where there were statistically significant differences between indie and non-indie testing. We clean up the data export from SurveyMonkey by transforming the Likert-like data into numerical values (1 = *Strongly Disagree*; 2 = *Disagree*; 3 = *Neither Agree nor Disagree*; 4 = *Agree*; 5 = *Strongly Agree*) for ease of analysis. We also encode the answers that are not in the Likert-like range (0 = *N/A*; -1 = *Unknkown*) for the purpose of counting how many of these responses we received per respondent and per question. For each of the Likert-like responses, we calculate the median and range overall for all responses and within-group for indies and non-indies. We compare the medians between indie and non-indie responses for each prompt to discover the magnitude and trend of differences between the groups for specific items. We also graph the responses to each prompt by group to examine the distributions. These distribution graphs can be seen in Figure E.1.

As the distributions are generally non-normal and the sample sizes are small (*n = 11* or fewer in each group), we perform Mann-Whitney U-tests [47], [49] on each set of responses to look for statistically significant differences in distributions that would illustrate a divergence between indie and non-indie testing. As *N/A* and *Unknown* responses are not relevant for the Mann-Whitney U-test, these values are omitted from the tests. The Mann-Whitney U-test is a non-parametric test that determines whether randomly-selected observations from the two populations being tested are likely to be equal (null hypothesis), or whether one is more probable to be greater than the other. There are differing methods for conducting the test and reporting its results [9], [20], so we describe our methods here.

We run our tests as two-tailed tests with $\alpha = 0.05$ to achieve 95% confi-

dence in our results. In accordance with the `asymptotic` parameter in the `mannwhitneyu()` function from the statistical package we use for the tests [74], we compare the U-statistic to a normal distribution with corrections for ties in ranks. We choose this approach because of its applicability to our data; the alternate `exact` option requires sample sizes smaller than 8 and does not make corrections for ties in ranks. Our sample sizes are larger than 8 and ties occur often in our data due to our five-item response scale. Bergmann also recommends using asymptotic approximation for sample sizes greater than 10 or samples containing ties [9]. We report the U-statistic from this test, which is the number of observations from one group that are greater than observations in the other in pairwise comparisons of samples. We always use indies as the first group, so U indicates how many indie observations are greater than non-indie observations. The maximum value of U is the product of sample sizes of the two groups, $n_1 n_2$, corresponding to the number of pairwise comparisons, and the closer U is to $n_1 n_2$, the more indies agree with the prompt in comparison to non-indies. We also report f, or the Common Language Effect Size (CLES) [49], expressed as a proportion of responses from the first group (indies) that are larger than the other. For example, a CLES of 75% means that in three-quarters of the pairwise comparisons, the indie value are greater than the non-indie value.

We plot our graphs using the *matplotlib* Python module and perform the Mann-Whitney U-tests with the *SciPy.stats* module. We compute the common language effect sizes manually and confirm using the *Pingouin* [98] module.

3.4.3 Qualitative Analysis

Our goal in qualitative analysis is to find out *what* the differences between indie and non-indie testing are and *how* they differ. We also want to learn what the groups had in common in terms of approaches, tasks, and concerns when it comes to development and testing, and where they are fundamentally dissimilar. For this, we use Reflexive Thematic Analysis (RTA) [11]–[13] to systematically code the open-ended text responses, which we gather into themes to analyze trends of items mentioned and their relationships.

Reflexive Thematic Analysis Approach

Reflexive thematic analysis is a qualitative analysis method for finding and analyzing themes and meanings that exist in the data while remaining cognizant of the analyst's role in interpreting it. It is used for several kinds of qualitative data, including interviews and surveys, and works with datasets of varying sizes. RTA places less emphasis on accuracy or reliability; unlike other qualitative methods such as content analysis, it is not the goal of RTA that two researchers produce the same codes and themes, and no inter-coder reliability measures are used as a result [13]. Instead, the focus is on a "reflective and thoughtful engagement with their data and their reflexive and thoughtful engagement with the analytic process" [12]. In other words, RTA acknowledges the researcher as an active participant in the process of constructing meaning from the data, and the researcher is encouraged to apply their domain knowledge and analytical skill through the process. As both researchers are long-time fans of video games, and one is an active commercial indie game developer, we find RTA to be the best-suited analysis for our circumstances. The way we collect the data using open-ended text responses through an online survey also contributes to this decision, as responses are limited to what the participant remembered and thought to mention. We are unable to follow up to ask clarifying or additional questions to respondents, and therefore treat responses less as a true and complete representation of how games are made and tested and more as recollections of parts of participants' specific experiences.

The process of reflexive thematic analysis involves a six-phase process. First, *familiarization* to become intimately familiar with the data and take initial notes as relevant. Second, we perform open coding for *generating initial codes* using brief but descriptive words and phrases to label parts of the data that are relevant to the research questions. This must be done in a fair and thorough manner with all data being given equal consideration. Once coding is complete, we move on to the third phase, *generating themes*, by finding commonalities between codes and grouping shared meanings together. In *reviewing themes*, the fourth phase, we review our themes to check whether they represent the data that comprise them and properly describe the overall dataset. For the fifth phase, *defining and naming*

themes, we give names and detailed definitions to each of the themes. *Producing the report* is the sixth and final phase, where the most important findings and analyses are reported along with illustrative extracts from the data. Although these steps are presented in order, RTA is not a linear process; iteration and recursion can occur at any phase, such as returning to code generation after reviewing themes on identifying codes which were not well-suited to the research questions [11], [12]. We use a recent worked example from one of the authors [13] as a guide for rigorously applying the process.

For reporting our RTA process and results, we provide literal extracts from participant responses in *"italicized quotation marks"*, codes in SMALL CAPS, and themes or subthemes in **bold text**.

Familiarization

We did not need to transcribe the data as responses were given in text format, but we did collate them separately in an Excel sheet from the rest of the data grouped by indies and non-indies. We read through all of the open text responses, highlighting certain key words and phrases from responses that are relevant to our research questions without formally coding them. This phase gave us a good idea of the content of the responses and the major items that were discussed. We read through the entire dataset twice in order to get a good idea of the full extent of the answers. As we read, we noticed that some responses were short, vague, or not particularly relevant. For example, some respondents wrote parts of their responses in imperative voice: *"Use Test kits that don't have the extra memory and PC configurations. Build hardware libraries to shared peripherals, monitor types, and graphic cards."* We also found that most responses did not prioritize capturing as much relevant information from the testing process as possible and offered brief insights into specific parts of testing. It was through this process that we decided to only code descriptions of testing experienced on the project, and to code what was explicitly stated or clearly implied. We did not want to assign meaning that the respondents may not have intended, such as interpreting the imperative statement above as a description of how testing was conducted on that project instead of as

general opinions and recommendations.

Generating Initial Codes

We then performed systematic coding of the text by going through all of the responses in order and identifying all descriptions and opinions of the respondent's experience with any aspect of testing on the project they chose. For our first pass, we coded on a prompt-by-prompt basis, tagging responses from all respondents for a prompt before moving on to the next prompt. For our second pass for verification and standardization, we coded on a respondent-by-respondent basis, coding all responses from one respondent before moving on to the next respondent. Since the level of meaning we were looking for was at the phrase level to describe QA processes, goals, tools, and outcomes, we coded using short phrases that represented the underlying description. We used an inductive approach to generate codes from the data itself instead of trying to force any preconceived notions or existing theoretical frameworks upon them. We were aware that several themes were suggested by the sections of the survey themselves, and the topical nature of the questions were likely to arise as themes. To avoid simply grouping responses into the respective question topic from which they were derived, all codes for all questions were grouped together during coding so that it was not evident which question's response the code came from.

We first highlighted all of the key words, phrases, and quotes, expanding upon the preparatory work done in the prior phase. Then, we summarized each highlight into its key points. For example, the sentence *"We had no funding, so us on the team just tested it during our 'dev time' using our Android devices we already owned"* was coded with NO FUNDING FOR TESTING, TESTING DONE BY TEAM ON OWN TIME, and USED ALREADY-OWNED HARDWARE (ANDROID) FOR TESTING. For specific types of tests and tools that were mentioned, we recorded in vivo codes, or codes formed from the respondents' own words, to reflect the actual terminology used by respondents. We performed coding separately for both non-indie and indie groups so that the items described in indie processes did not accidentally end up in the non-indie group. We performed coding manually using Excel.

Generating Themes

We then examined the codes to look for similarities and broader themes they may belong to. We analyzed the indie and non-indie groups separately without assuming that the same themes would arise, as differences in responses between the groups may lead to differences in themes. We gathered all of the codes that we identified in the Excel sheet into a mind map software tool as a flat structure with all of the codes being first-order items from the group in question as the root. We connected related codes, such as one code describing a testing process and another describing sentiments about that process, with a dotted line to preserve relationships. We identified similar codes and reorganized them in proximity to one another; as we noticed commonalities in codes, we created brief headings to represent those common groupings and moved the relevant codes under them. We repeated this process until all of the codes were accounted for under a group. We made the conscious decision in this process to make automation a priority category as it was one of the primary areas of interest we wanted to analyze for differences between the groups. This meant that if there was a code relating to goals for automated testing, it was placed into the automation category instead of the goals category, as automation took priority.

Once all of the codes were categorized, we analyzed codes within each category for sub-categories using a similar approach to the initial categorization. We repeated this process until no more sub-categories could be identified. For example, the category **Resources** included subcategories **Human Resources**, **Hardware**, **Time**, **Budget**, and **Resources Sentiment**, and **Human Resources** had further sub-categories for internal and external human resources. Our goal was to have at least two codes per category, but we allowed categories with only one code if the contents of the code were important enough to warrant their own branch, such as in the case of a single **Test Plan Sentiment**. We had to return to code generation several times during this process to make sure we had identified and worded codes consistently, and to double-check the context of some codes to make sure that we categorized them correctly. The categories that we defined for indies and non-indies ended up being very similar, so we standardized the categories and

subcategories between the groups for ease of comparison. The categories that we defined in this step became our themes.

Reviewing Themes

Following theme generation, we reviewed the themes to ensure that they were in alignment with the codes that we assigned as well as overall dataset. We checked that themes were properly assigned (e.g., none of the themes should be codes instead), whether the themes revealed insights related to our research questions, what the inclusion and exclusion criteria were, and that the codes adequately supported their themes. Two different team members reviewed the notes and codes separately for coherence and consistency.

Theme Definition and Reporting

Once the themes and codes were reviewed and set, we refined the definition of each theme to be clear about what was being described in the data and that the theme name properly fit. We then analyzed the codes within each theme to understand what was being said about that theme by the respondents in the group and compared our findings between the groups. We also compared our process with Braun and Clarke's checklist for proper thematic analysis [11] to make sure our procedure and analyses were trustworthy. Finally, we report our findings in this thesis with examples from responses and recommendations for improving quality assurance practices and tools for resource-constrained game developers.

3.5 Summary

In this chapter, we introduced our online survey of commercial game developers. We discussed the details of the survey, including our pilot survey and the changes we made from the feedback we received. We explained how we recruited participants, validated and grouped our data, and performed our analyses. Namely, we conducted quantitative analysis using Mann-Whitney U-tests to find statistically significant differences of response distributions between indies and non-indies and

qualitative analysis using reflexive thematic analysis to identify themes from the open-ended text responses.

Chapter 4

Survey Results

4.1 Respondent Information

Respondents have a median of 6 years of experience in game development. Figure 4.1 is a boxplot of overall participant experience in years, with "Game Dev" being overall game development experience and "Indie", "AA", and "AAA" being years of experience in that group in the past ten years. Over the past ten years, 15 respondents had indie development experience, 7 had AA development experience, and 10 had AAA development experience. We asked participants what roles they had held in game development over the course of their career. The choices available were:

- Quality Assurance and/or Game Tester ("QA")
- Developer, Programmer, and/or Engineer ("Developer")
- Game Designer, Level Designer, Gameplay Designer, and/or Product Owner ("Designer")
- Team Lead, Producer, and/or Project Manager ("Lead")
- Other (specify)

Figure 4.2 shows the breakdown of roles held by respondents in the past. Lead, Designer, and QA roles each have 11 participants (57.9%) who held the role in the past, with 7 respondents (36.8%) holding a Dev role. Four (21.1%) respondents

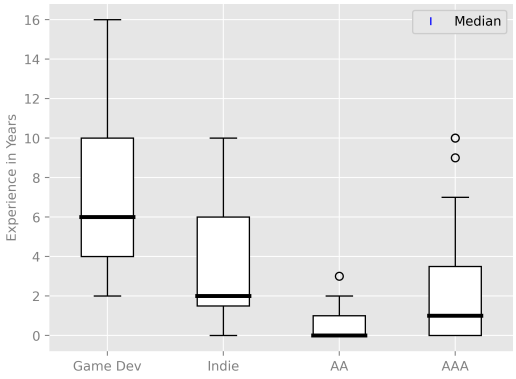


Figure 4.1: Boxplot of respondents' experience in game development. "Game Dev" is overall experience, "Indie/AA/AAA" is experience in the past 10 years.

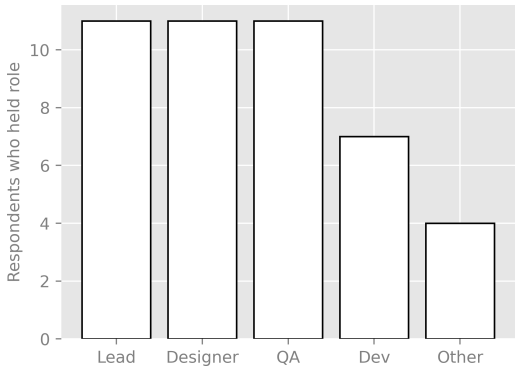


Figure 4.2: Roles that respondents held over their careers.

wrote in entries for *Other* roles, which are *UX Researcher, Business/Marketing, Engineering Manager, Release Manager, LiveOps Manager, and LiveOps Program Manager*. The total of roles held or other entries submitted do not add up to the number of respondents as many participants held multiple roles. All of the write-in roles are additional to other roles that the respondent held with the exception of *UX Researcher*, which is the only role that respondent R17 had held in industry. Regardless of the role they held, all participants had experience with testing for commercial games as described in their open text responses.

Thirteen respondents (68.4%) are between 25 to 34 years of age, and six respondents (31.6%) are between 35 to 44 years of age. Canada is the top country of residence with 13 responses (68.4%), followed by the United States with 4 (21%), the United Kingdom with 1 (5.3%), and the Netherlands with 1 (5.3%). Overall, participants have a breadth of experience in different roles and project scopes, and over half of them have experience in a QA role. The full table of participant information broken down by respondent is available in Appendix C.

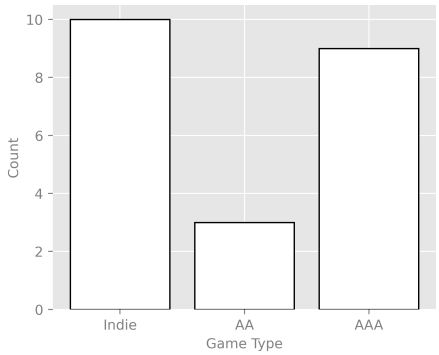


Figure 4.3: Game submissions by type.

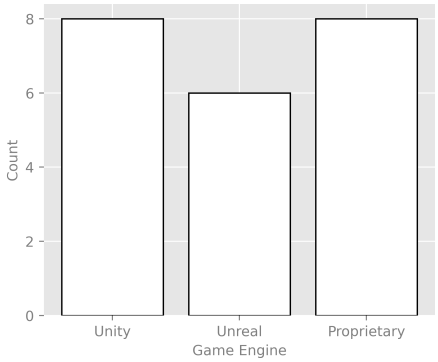


Figure 4.4: Game engines used on submitted projects.

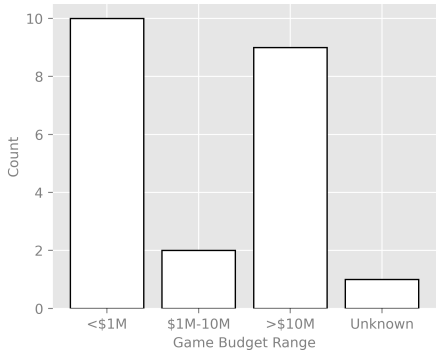


Figure 4.5: Budget ranges for submitted games.

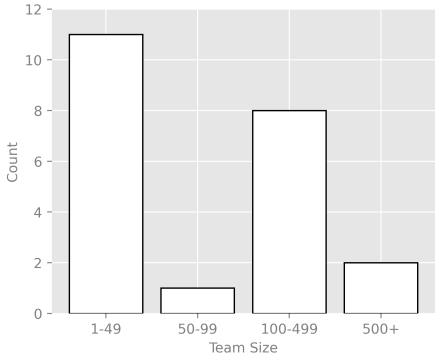


Figure 4.6: Team size ranges for submitted games.

4.2 Project-based Responses

We received 22 project submissions through the survey. Figure 4.3 breaks down the games by type, with 10 indie games (45.5%), 3 AA games (13.6%), and 9 AAA games (40.9%). The project-based responses for these games show many differences between the groups that are indicative of their relative resource availability, which we use as factors in dividing the dataset. Figure 4.4 shows the comparison between engines used: 8 projects use the Unity engine [95] (36.4%), 6 projects use Unreal Engine [97] (27.3%), 8 projects use Proprietary or Custom engines (36.4%). Of the proprietary engines used, 2 projects specify the Frostbite engine [28] (9.0%).

Figure 4.5 shows the relative project budget ranges for these games: 10 projects have a budget under $1M USD (45.5%), 2 have between $1M–$10M USD (9.1%), 9 have over $10M USD (40.9%), and 1 is unknown (4.5%). Team sizes are summarized

Table 4.1: Project information for indie games.

Game ID	Game Type	Game Engine	Team Size	Game Budget	Publisher Support	Project Duration
G1-I	Indie	Proprietary	1–49	<$1M	Yes, without marketing	3 years
G4-I	Indie	Proprietary	1–49	<$1M	Yes, with marketing	3 years
G5-I	Indie	Unity	1–49	<$1M	No	1 year
G8-I	Indie	Unity	1–49	<$1M	No	3 years
G9-I	Indie	Unity	1–49	<$1M	No	3 years
G11-I	Indie	Unity	1–49	<$1M	No	1 year
G16-I	Indie	Unity	1–49	<$1M	No	3 years
G17-I	Indie	Unity	1–49	<$1M	No	2–3 years
G19-I	Indie	Proprietary	1–49	<$1M	Yes, with marketing	2–3 years
G20-I	Indie	Unity	1–49	<$1M	No	3 years
G21-I	Indie	Proprietary	1–49	$1–$10M	No	1 year

Table 4.2: Project information for non-indie games.

Game ID	Game Type	Game Engine	Team Size	Game Budget	Publisher Support	Project Duration
G2-N	Non-Indie	Frostbite	100–499	>$10M	Yes, with marketing	3 years
G3-N	Non-Indie	Unreal	50–99	>$10M	Yes, with marketing	2–3 years
G6-N	Non-Indie	Unreal	100–499	>$10M	Yes, with marketing	3 years
G7-N	Non-Indie	Proprietary	100–499	>$10M	Yes, with marketing	3 years
G10-N	Non-Indie	Proprietary	500	>$10M	Yes, with marketing	3 years
G12-N	Non-Indie	Frostbite	500	>$10M	Yes, with marketing	3 years
G13-N	Non-Indie	Unreal	100–499	>$10M	Yes, with marketing	3 years
G14-N	Non-Indie	Unreal	100–499	$1M–$10M	No	2–3 years
G15-N	Non-Indie	Unity	100–499	>$10M	No	3 years
G18-N	Non-Indie	Unreal	100–499	Unknown	Yes, without marketing	3 years
G22-N	Non-Indie	Unreal	100–499	>$10M	Yes, without marketing	3 years

in Figure 4.6—11 projects have 1–49 team members (50%); 1 have 50–99 members (4.5%); 8 have between 100–499 members (36.4%); and 2 projects have over 500 members (9.1%). Since budget, team size, and publisher/marketing support are used as our criteria for classification, we expect that results for these fields would be relatively consistent, which is the case and can be seen in the similarities between Figure 4.3, Figure 4.5, and Figure 4.6. The full table of project information is available in Appendix D.

4.3 Data Organization

Given the low number of AA projects submitted as well as our focus on differences between indie and non-indie developers, we divided our responses into two

groups: indie and non-indie. Games reported as indie were put into the indie group, and games reported as AAA were put into the non-indie group. We also reallocated the 3 AA projects to these two categories. The 3 AA games, G14, G18, and G21, state having budgets between $1M–$10M USD and no publisher support. As G14 and G18 have between 100–499 members on its team, we classify it as a non-indie game as we expect that the processes and tools for a team of this size would be more in line with AAA practices. G21 has 1–49 team members and was classified as an indie game alongside the other projects self-reported as indie games. Project identifiers will be suffixed to indicate whether they are indie (I) or non-indie (N) games, such as the indie game G21-I or non-indie game G18-N.

After the division into indie and non-indie games, we have 11 indie and 11 non-indie games for consideration. Figure 4.1 and Figure 4.2 contain the key project information for each group. Of the 11 games in the indie group, 7 use Unity (63.6%) and 4 use proprietary engines (36.4%). All 11 teams have between 1–49 members, and 10 projects have budgets under $1M USD (90.9%). Only 2 indie projects have a publisher who provided marketing support (18.2%); 1 has a publisher without marketing support (9.1%) and 8 have no publisher at all (72.7%).

In contrast, there are 6 non-indie games that use Unreal Engine (54.5%), 4 games that use proprietary engines, including the 2 who specify the Frostbite engine (36.4%), and 1 that uses Unity (9.1%). Eight of the teams have between 100–499 people (72.7%), 2 teams have 500 or more (18.2%), and 1 has 50–99 people (9.1%). The majority of projects have over $10M USD in funding, with 9 games (81.8%) in this range; 1 game has a budget between $1–$10M USD, and 1 game's budget range is unknown to the respondent. Seven of the non-indie projects have a publisher who provided marketing support (63.6%), 2 have a publisher without marketing support (18.2%), and 2 have no publisher attached (18.2%).

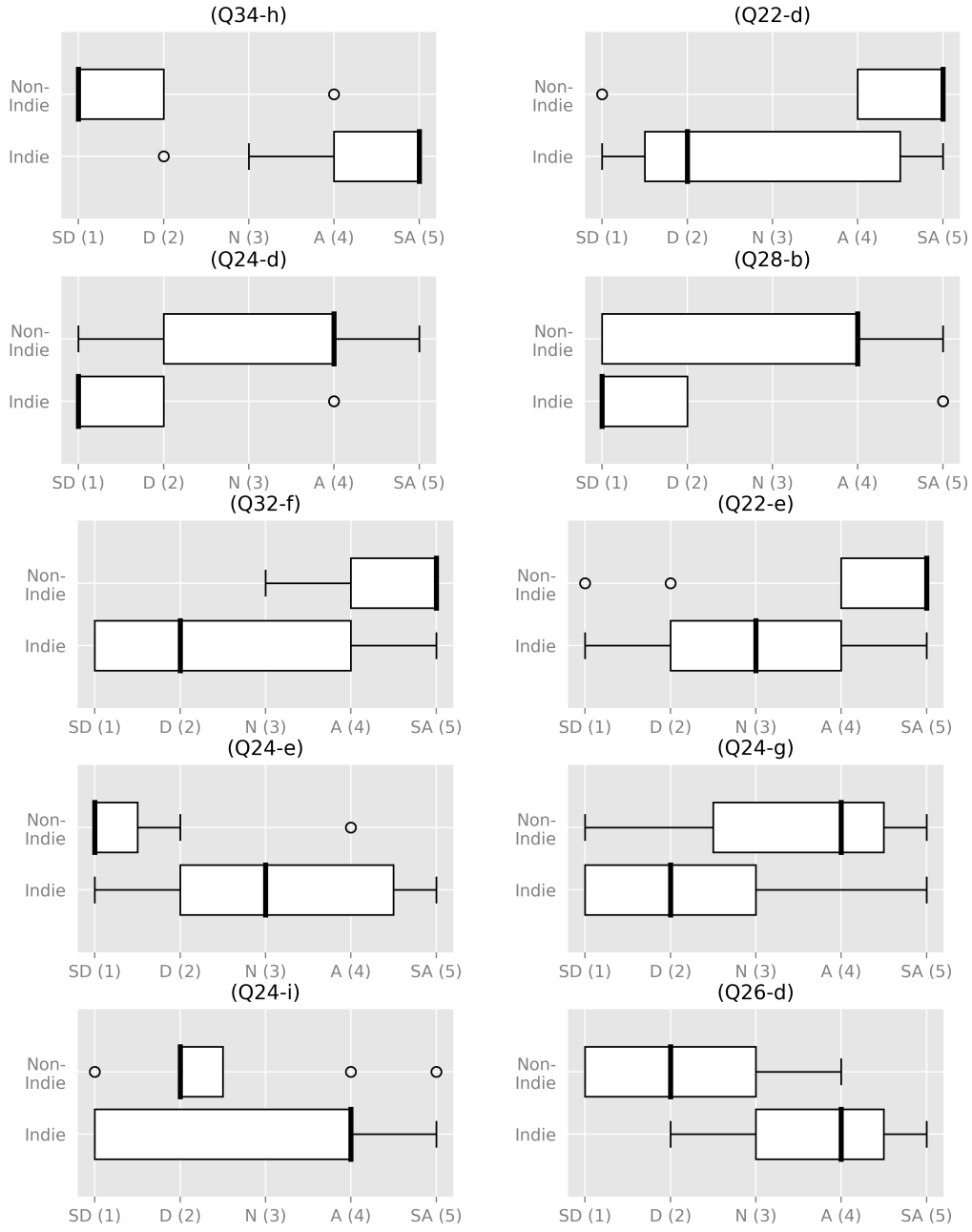


Figure 4.7: Boxplots of responses for the top ten differences in medians. The text of the corresponding prompts are shown in Table 4.3.

Table 4.3: Differences in Likert-like response medians between indies and non-indies in decreasing order. A positive delta indicates stronger agreement from non-indies than indies, and vice-versa.

Prompt ID	Delta	Prompt Category	Prompt Text
(Q22-d)	3	Performance	The majority of testing was done by a dedicated QA team
(Q24-d)	3	Planning	Each major feature for development had a testing or validation plan prior to implementation
(Q28-b)	3	Automation	We had personnel dedicated to writing scripted or automated tests on this project
(Q32-f)	3	Results	The results of testing were monitored or visualized over time and used as metrics
(Q22-e)	2	Performance	We routinely ran unit tests on this project
(Q24-g)	2	Planning	Test plans and priorities were primarily set by quality assurance personnel
(Q28-g)	2	Automation	Automated testing yielded results that could not be determined through manual testing
(Q34-b)	2	Resources	We had sufficient testing/QA personnel dedicated to this project
(Q34-d)	2	Resources	We had appropriate testing tools given the project and its testing goals
(Q24-e)	-2	Planning	There were no concrete testing plans or priorities for the majority of this project
(Q24-i)	-2	Planning	Test plans or priorities were primarily set by the producer, game director, or senior management
(Q26-d)	-2	Goals	The results of testing were often subjective and open to interpretation
(Q32-e)	-2	Results	The results of testing were used primarily by leads and managers
(Q34-h)	-4	Resources	Testing was done primarily by team members who had other primary tasks

4.4 Quantitative Differences between Indie and Non-Indie Responses

4.4.1 Comparing Indie and Non-Indie Response Medians

We compare each set of responses from indie and non-indie developers by question. We first find the response median per prompt for each group and compare them for differences, with a difference of two or more being considered significant as it signifies a shift in overall opinion (e.g., from Disagree (2) to Agree (4)). By comparing medians ($\Delta = median_{nonindie} - median_{indie}$), we find four prompts with a difference of 3, five with a difference of 2, four with a difference of -2, and one difference of -4. Table 4.3 shows the prompts and deltas, where a positive delta signifies that non-indies express more agreement with the prompt than indies (and vice-versa). Figure 4.7 contains boxplots for the first 10 deltas to visualize the medians and spreads (Figure E.1 contains boxplot results for all questions). (Q34-h) has the highest difference between groups, indies agreeing far more strongly than non-indies that testing is mainly performed by members who had other primary tasks, such as developers or designers. For all of the next four-highest differences, non-indies agree more than indies that testing was done by a dedicated QA team, each major feature had a testing plan prior to implementation, there were people dedicated to writing and maintaining automated tests, and results from testing were tracked or visualized over time and used as metrics.

Finding these deltas gave us an idea of where there were different tendencies between indie and non-indie projects, and which way the differences leaned for indie developers. Appendix F contains the boxplots of responses for all prompts.

4.4.2 Comparing Indie and Non-Indie Response Distributions

We then want to see which of these differences in medians or distributions were statistically significant. For this, we use the two-tailed Mann-Whitney U-test to check the prompts for which the distribution of responses in the two groups are significantly different. We choose this test due to its robustness with non-normally distributed data from small sample sizes and ability to handle ordinal data. The data are independent as each project is only classified as indie or non-indie with no overlap. The test is two-tailed because we are examining for general equality of distributions and not whether one group is greater than the other. The responses being measured are ordinal Likert-like data converted to a numerical scale of 1 to 5. For our two-tailed Mann-Whitney U-tests, we set $\alpha = 0.05$ and our hypotheses are:

- H_0: the distributions of both populations are equal
- H_A: the distributions of both populations are not equal

A p-value result under $\alpha = 0.05$ constitutes a statistically significant difference between indie and non-indie testing for that prompt with 95% confidence.

Table 4.4 shows the 11 prompts for which we reject the null hypothesis, as well as the prompt's topic, median and sample sizes for each group ($n < 11$ signifies that there are "N/A" or "Unknown" responses for that prompt that were discarded for the analysis purposes), the test statistic, and the CLES. The prompts identified come from the Planning, Goals, Results, and Resources categories, with no statistically significant differences discovered for Performance, Automation, and Tools categories. The remainder do not have sufficient proof to reject the null hypothesis. The full results from the Mann-Whitney U-tests are available in the study artifacts [18].

Table 4.4: Prompts with significant differences ($\alpha = 0.025$) in responses between indies and non-indies. A higher U-statistic or CLES indicates stronger agreement from indies than non-indies, and vice-versa. Sample size is 11 for all samples except medians marked with an asterisk* to indicate a sample size of 10 or a dagger† for a sample size of 9.

Prompt ID	Prompt Category	Prompt Text	Indie Median	Non-Indie Median	Mann-Whitney U-Statistic	P-Value	Effect Size
(Q24-d)	Planning	Each major feature had a test plan prior to implementation	1	3†	18.5	0.0153	18.7%
(Q24-e)	Planning	There were no concrete testing plans or priorities for the majority of this project	3	1	100.0	0.0068	82.6%
(Q24-g)	Planning	Test plans and priorities were set by QA personnel	2	4	30.5	0.0477	25.0%
(Q26-d)	Goals	Testing results were subjective and open to interpretation	4	2	100.5	0.0080	83.1%
(Q32-c)	Results	Testing results were used to validate internal targets	4	4*	25.0	0.0207	22.7%
(Q32-f)	Results	Testing results were monitored or visualized over time as metrics	1†	5*	17.0	0.0184	18.9%
(Q34-b)	Resources	We had sufficient testing personnel on this project	2	4	18.0	0.0040	14.9%
(Q34-c)	Resources	We had sufficient automated testing on this project	2†	3*	19.0	0.0286	21.0%
(Q34-d)	Resources	We had appropriate testing tools for this project and goals	2	4*	9.5	0.0010	8.6%
(Q34-g)	Resources	Testing was primarily performed internally on the team	5	4	93.5	0.0061	77.3%
(Q34-h)	Resources	Testing was done primarily by team members who had other primary roles	5	1	116.0	0.0002	95.9%

Ranked by order of common language effect size, or the probability that an observation from the indie group is higher in agreement than an observation from the non-indie group, the most significant finding from the tests is that 95.9% of indies are more likely to agree than non-indies that testing is done primarily by team members who have other primary roles; in other words, that quality assurance for indies is a secondary or additional role for the majority of indie developers. We also find that 91.4% of non-indies are more likely to agree that they had adequate testing tools on their project, implying that the vast majority of indies feel that they did not have adequate QA tooling available to them. 83.1% of indies are more likely to agree that test results were subjective and open to interpretation, and 82.6% of indies are more likely to agree that there were no concrete testing plans or priorities on their project.

Table 4.5: Six projects with the highest opt-out responses in descending number of opt-outs.

Project ID	Number of Opt-Outs	Top Opt-Out Category	Opt-Outs in Top Category
G18-N	22 Unknowns	Automation	10
G2-N	16 Unknowns	Automation	9
G16-I	15 N/As	Automation	8
G12-N	13 Unknowns	Automation	5
G11-I	11 N/As	Automation	6
G20-I	10 N/As	Automation	6

4.4.3 "Not Applicable" and "Unknown" Responses

We examine the frequency and distribution of "Not Applicable" ("N/A") and "Unknown" responses in our data, which are excluded from our Mann-Whitney U-tests. We refer to these responses as as opt-outs, since selecting one of these options opts the respondent out of answering on the five-point scale. We look at the statistics for opt-out responses on a project-by-project as well as a prompt-by-prompt basis to see where there are gaps in testing usage or awareness. We interpret "N/A" to mean that the focus of the prompt was not relevant to the project, such as questions about automation for a project where automation was not used, and "Unknown" to mean that the focus of the prompt may have applied to the project but the respondent was unaware of the particulars and did not know enough to make an informed response.

We report the top six occurrences of opt-outs in Figure 4.5 along with the highest opt-out category. In all six cases, the category with the highest number of opt-out responses is Automation, making it the aspect of testing that is least applicable and least known about overall. The second-highest category containing opt-outs is Tools. No other project responses had a combined number of opt-outs totalling 10 or more. For the three highest "N/A" responses by project, all three projects described are indie games, and the open-ended text response for these respondents indicates that testing on their project was entirely manual with no automation. The top "Unknown" responses are all from AAA projects; for G2-N, the respondent knows there was automated testing done on the project but was

not privy to details, and for G18-N and G12-N, the respondent was a QA tester who did not use automation for their area of responsibility but believed automation was used elsewhere on the project. The median for the count of each opt-out response type from all projects is 0.5, indicating a low overall rate of "N/A" and "Unknown" responses.

On a prompt-by-prompt basis, the prompts with the highest number of combined opt-outs are (Q28-c), *The automated testing tools that we used required frequent adjustments as the project progressed*, with 7 "N/A" and "3 Unknown" responses and (Q28-j), *Automated testing was a source of frustration on this project*, with 6 "N/A" and 4 "Unknown" responses. The prompts with the second-highest number of combined opt-outs are (Q28-e), *Automated testing yielded more actionable results than manual testing*, and (Q28-g), *Automated testing yielded results that could not be determined through manual testing*, each with 5 "N/A" and 4 "Unknown" responses. The median count of "N/A" responses per prompt is 0 and the median count of "Unknown" responses per prompt is 1, meaning most questions did not have opt-outs.

4.5 Qualitative Analysis

We identify ten primary categories through our reflexive thematic analysis: testing approach, goals, plans, types, tools, resources, timing, automation, results, and overall sentiment. Most of these themes reflect the categories of our questions. We do not identify a single theme relating to our survey section on test performance, since we found that responses relating to test performance are better represented by **Test Types**, **Testing Approach**, and **Test Timing**, depending on what the focus of the code is. We discuss our major findings from each theme below.

4.5.1 Testing Approach

Testing Approach thematically covers items relating to testing philosophies, policies, and processes. They answer the question, "How was testing done on the project?" The sub-themes are **testing philosophy**, focused on the general approaches the team took to QA; **testing procedures**, or what testing was done;

test responsibilities for any specific responsibilities that were defined; and **approach sentiment** for any opinions or outcomes expressed related to the theme.

> “Integration test plans were handled by the product owner, while plans for unit testing were handled by the developers. The developers were also given the opportunity to request specific focus for the integration tests if they were concerned about a specific set of interactions”
>
> *-G8-I (Indie)*

> “QA/Testing was mostly a fluent process and not necessarily structured. As a small team it didn’t make sense to invest the resources to do so.”
>
> *-G4-I (Indie)*

For testing philosophies, indies describe a largely chaotic process with codes such as Testing not structured, No formal testing plans, and ad-hoc testing without plans. Non-indies by comparison describe complex and well-planned approaches, such as test plans made early in project, stringent technical testing, ability to release at any time, and each system could be tested in isolation. These codes, paired with the fact that there are 5 responses for this subtheme from indies and 16 from non-indies, indicate that indies may not consider their approach or values to testing as deeply as non-indies do.

Indies describe procedures that are primarily concerned with playtesting, such as tested by playing entire game, tested by playing individual sections, observed players interacting with game, got player feedback at demos and events, and recorded test sessions live on Twitch, although one code states that they did not solicit player feedback. Comparatively, non-indies focus on their formal procedures and pipelines, such as having their releases staged through a separate testing product, systematically testing through a checklist of in-game tasks or testing on all supported platforms, formal bug report templates that everyone adhered to, and communication of validation criteria.

> “As a GaaS project, we had to take a pretty divergent path from traditional game dev when it came to testing and pushed for a Quality at Source approach to building and testing the game - meaning that we were always testing from day one as part of a continuous integration strategy and not relying on long testing periods at the end of each milestone to secure quality builds”
>
> *-G13-N (Non-Indie)*

> “Everything and every system had an isolated test level and the ability to test the system in isolation with known results and assets. This aided root cause analysis and rule out core functionality errors from more compounding issues.”
>
> *-G3-N (Non-Indie)*

Indies' responsibilities for testing generally fall to the whole team with developers responsible for thoroughly testing their own features before integrating (DEVELOPER RESPONSIBLE FOR TESTING FEATURES BEFORE INTEGRATION is mentioned in 3 projects), and the product owner performs build checks and thorough tests. Non-indies have more access to dedicated QA staff with specific responsibilities (e.g., QA PERFORMED REGRESSION TESTS mentioned in 3 projects), and testing is more defined (e.g., DEVS WROTE UNIT TESTS ALONG WITH FEATURES).

For sentiments about overall testing approaches and procedures, indies have no relevant codes, whereas non-indies overall paint a picture of a successful but constantly changing process. They feel that they contributed to a WILDLY SUCCESSFUL PROJECT, that TESTING WAS SUCCESSFUL WITH MINIMAL GAME-BREAKING BUGS AT LAUNCH, and TESTING SYSTEMS IN ISOLATION HELPED IDENTIFY ROOT CAUSES despite also feeling that the PRODUCTION PROCESS WAS HIGHLY VOLATILE and the TESTING PROCESS REQUIRED A LOT OF FINE-TUNING.

Overall, indies describe their approach and process as chaotic, unstructured, and focused on user experience while non-indies delineate a systematic approach with technical pipelines and resources not reflected in indies' experiences. We

believe that the dedicated QA positions in non-indie development affords those testers the time and space to focus on testing philosophies and processes, bolstered by their access to resources for testing, whereas indie testing as described by respondents was a secondary task for members whose time was primarily dedicated to other work.

4.5.2 Testing Goals

Testing goals describe the aims for different types of testing that were done on the project and address the question, "Why was testing performed, and what was it trying to discover?"

> "With our limited resources, the main priority was game-breaking bugs. To our knowledge, no version shipped with any game-breaking bugs, so it seems we were successful here."
>
> *-G16-I (Indie)*

> "Later on in the project lifecycle, testing turned toward user experience. This was accomplished by sending builds to a dedicated audience for early-access review."
>
> *-G8-I (Indie)*

Indies' primary stated goal is FINDING BUGS, which is mentioned in 5 projects. Some of the bugs they mention include GAME BREAKING BUGS, DATA SYNC ISSUES, and OUT OF BOUNDS ISSUES. Four projects mention USER EXPERIENCE as an explicit goal, and 2 mention BALANCE ISSUES. Most of the goals stated by indies do not contain detail and are analogous to a type of test. For example, goals such as REGRESSIONS, CONSOLE CERTIFICATION, and INTEGRATION are simply rephrasings of regression tests, console certification tests, and integration tests without additional details.

> “New features function as expected. Prior features continue to function as expected. Performance maintains within expected levels across supported devices. Infrastructure is able to support anticipated load.”
>
> *-G15-N (Non-Indie)*

> “We would be testing out the build to see if there were any regressions, bugs, or any new issues that came with new features that were being implemented to stay on top of issues with the build... with the external playtests, we were looking for player attitudes and seeing if players were enjoying the game, understanding it, and if there were any issues they were running into that we were not aware of.”
>
> *-G18-N (Non-Indie)*

For non-indies, REGRESSIONS and FUNCTIONALITY are mentioned the most with 3 each, followed by 2 mentions of BLOCKING ISSUES. Non-indies mention more aspects of testing goals, such as who set goals (e.g., GOALS SET BY PRODUCTION TEAM), as well as specific goals for specific tests (e.g., LOCALIZATION TESTING FOR HOOKUP AND FORMATTING, FOCUS TESTING FOR PLAYER BEHAVIORS). They also describe changes to goals over the course of the project, such as basic functionality and stability being a focus in early development that changed to cosmetics and edge cases in late development. Their descriptions of goals are also more detailed; whereas indies simply state USER EXPERIENCE, non-indies describe focus testing for player enjoyment, attitudes, understanding, and subjective feedback. One respondent describes a non-indie project where the priority became shipping the game, with design becoming an afterthought. This change in priorities led to their testing goals focusing on sanity tests and eliminating blocking bugs. This case highlights the commercial nature of games, where getting a product to market often takes precedence over guaranteeing product quality, especially as a function of the costs involved in AAA game development. While general software development is also affected by economic realities, games place a much higher emphasis on the core requirement, “fun” [39], [53], [69], and failing to prioritize this require-

ment for testing can lead to the game floundering on the market.

Non-indie goals are more numerous, detailed, and comprehensive compared to indie goals. The lack of specificity from indies may be indicative of a lack of clear goals and goal-setting in indie game testing. Combined with the previously-discovered focus on playtesting, we believe that indies are performing more ad-hoc testing to find bugs as they arise instead of testing with a systematic framework with clear priorities.

4.5.3 Test Plans

In this theme are codes relating to making plans for testing, including who was responsible for planning, who had input into the plans, and how plans were changed or maintained.

> “There were no concrete test plans. Testing was done in an ad hoc manner by developers and designers as the project progressed.”
>
> *-G8-I (Indie)*

> “We made no test plans. We were a small team and were each responsible for testing our own content. This was not done through a formal process.”
>
> *-G20-I (Indie)*

Multiple indies say they had NO TEST PLANS on their project, which agrees with the ad-hoc approach and goals we discovered. Those who do have test plans describe processes that are similar to non-indies: both describe testing features based on request from team members and QA and leads making test plans. Non-indies describe the types of tests they plan, how those plans are changed during the project, and how they are maintained, which are not mentioned by indies. Indies are more descriptive about how plans and requirements are generated, such as being drawn from a game design document or being made by the developers and submitted to QA with a build.

> “By having a very regular and consistent testing schedule, it helped us find any bugs or issues early on, and also to help us make sure that everything was on track for our desired game experience”
>
> *-G18-N (Non-Indie)*

> “Test plans made and maintained by internal QA in test rails. Guidance on expected behavior of features provided by the features primary designer. Engineering group separately implemented unit and regression tests to their understanding of the feature.”
>
> *-G15-N (Non-Indie)*

Non-indies describe a more formal and deliberate process in planning and demonstrate understanding of who is responsible for various stages of the process but less knowledge about how tests and requirements are put together compared to indies. Since non-indie projects report more QA testers and resources, it is reasonable that they would have well-defined and documented processes for plans without as much visibility into the reasoning behind them, whereas indies are more likely to be directly exposed to test planning processes for their game due to smaller teams and multiple responsibilities. These findings echo our quantitative results which found that indies do not have test plans for their features prior to implementation and do not have concrete testing plans or priorities for the majority of the project.

4.5.4 Test Types

Test types are a straightforward theme encompassing the various tests mentioned in the responses. Codes in this theme answer the question, “What types of tests were performed on this project?”

> “Basic testing on new features was done by developers and then passed to the QA department. The QA department would run smoke tests

every morning and integration and regression tests on new features that came in.”

-G20-I (Indie)

“We had a full QA team through the publisher that would regularly do Smoke, Regression, and CQA tests.”

-G19-I (Indie)

Regression tests are mentioned the most by both indies and non-indies, highlighting their importance in the development cycle in both groups. Indies have far fewer codes in this theme compared to non-indies; the majority of items mentioned by indies are also mentioned by non-indies. Some items mentioned by indies but not non-indies include BALANCE TESTS, EARLY ACCESS, and CODE REVIEW. Indies also mention playtesting multiple times. Non-indies list more types of tests covering more aspects of development, including ALPHA TESTS, BOOTFLOW TESTS, CORE PATH TESTS, LOCALIZATION TESTS, and VOICEOVER TESTS. They also discuss functional and non-functional testing, whereas indies do not. Neither group mention soak tests, although non-indies do refer to stress tests.

“Limited unit tests were performed as part of continuous integration. The rest of the testing was functional and non-functional manual grey-box (assisted by debug tools), exploratory testing, confirmation testing (regressing the fixed bugs).”

-G7-N (Non-Indie)

“Divided into Tech (Functional) and Content (Qualitative) Test teams. There were aspects of test case management for regression and session based testing for qualitative. All checks were tracked where Functional had elements of Unit, smoke that also aided in governing when qualitative perform exploratory and ad-hoc testing.”

-G3-N (Non-Indie)

In sentiments about different types of tests, indies state that unit testing is difficult to use for determining user experience issues, ineffective for transactional programming methodologies, and requires resources they do not have despite being valuable to the project. Non-indies also mention a lack of unit testing. They also feel that functional and automated testing fail to find exploratory issues and that black box testing is not useful for them.

From these results, we propose that indies either perform fewer types of tests than non-indies, or do not know about as many types of tests. Indies also focused more on lower-level or manual types of testing, such as code reviews, beta tests with external players, and manual build verification. Some of the test types not mentioned by indies may also indicate differences in capabilities and content between the groups, such as a lack of localization tests, voiceover tests, and side quest tests simply pointing to a lack of localization, voiceovers, or sidequests in the indie games our respondents developed.

4.5.5 Testing Tools

Testing tools cover any tools, primarily software, that were used to aid in the testing process. This includes tools that perform testing as well as those that help create, manage, or document them.

> “We used Trello for tracking issues and their progress and that’s it.”
>
> *-G4-I (Indie)*

> “We used Unity Cloud Build to provide up-to-date builds. The major benefit to this was that build issues were found immediately, while they were easy to diagnose... Unity Cloud Diagnostics was also used once the project was given to a random assortment of users in early access. The major benefit here was getting automated, detailed reports of issues that otherwise wouldn’t have been possible with manual reporting from this population.”
>
> *-G9-I (Indie)*

Many indies state that they use NO TESTING TOOLS on their project. A handful of tools are mentioned, including two first-party tools (IN-HOUSE GAME STATE EDITOR, TOOLS FOR PROJECT VALIDATION) and a few third-party tools. Many of the third-party tools mentioned are general-purpose freely accessible software, such as Discord [25], a messaging and community platform; Git Large File Storage [31] for large assets; NUnit [56], a free and open-source .Net unit testing framework integrated into the Unity engine; Redmine [65], a free and open-source project management webapp; Trello [90], a project management software that can be used without a paid subscription; Twitch [92], a popular video streaming platform; and Unity Profiler [96], a built-in performance profiling tool. Jira [6], a project management tool well-suited for Agile methodologies, can also be used without a paid subscription for up to ten users with limitations. Unity Cloud Build [93] and Unity Cloud Diagnostics [94] are paid subscription services.

> “We used several in house built tools, all wrapped into one software package. This software allowed us to log crashes (and their frequency), as well as record sessions, record performance across sessions, file bugs and automatically attach logs/screenshots.”
>
> *-G12-N (Non-Indie)*

> “A lot of in-house tools. However, most tools were either ‘This is old and doesn’t work that well’ or ‘This is new and it COULD be cool if it worked’. Documentation tools were completely neglected.”
>
> *-G2-N (Non-Indie)*

None of the non-indies state that they do not use testing tools. They list a number of paid third-party tools, including Azure DevOps [7], DevSuite [23], DevTrack [24], Team City [86], and Test Rail [87], but do not go into detail about what they are used for or how they are integrated into testing processes. Non-indies discuss first-party tools in more depth and detail, describing tools for filing bugs with automatically attached logs and screenshots, recording play sessions and comparing performance across sessions, in-game bug identification and marking, and de-

ploying builds to QA. Non-indies have access to many paid tools and have the resources and capacity to develop first-party tooling with complex functionality. These tools are generally considered helpful and useful, such as the automated bug submission tool which they describe as easier than teaching users to find the appropriate logs, take helpful screenshots, and attach them to bug reports. However, they state concerns regarding some of the tools being old and out of date. Additionally, some documentation tools are neglected, and first-party tools may stay broken for a long time if anything goes wrong with them, exposing a lack of maintenance and upkeep.

The tools described by indies point to a need for resourcefulness and creativity to make the best of free tools for their testing purposes. The limited paid automated tools they mention are reviewed positively, as they find issues quickly and provide detailed reports not possible through manual testing. They state frustrations with NUnit lacking support for the types of testing they want. Since indies had loosely defined goals for testing, and testing tools should be selected for their fit to those goals, it may be difficult for indies to select or develop appropriate and effective tools without first clearly defining their aims. This confirms our quantitative finding that indies did not feel that they had the appropriate testing tools to fit their project and goals.

4.5.6 Testing Resources

In this theme, we look at resources available to the project with **Human Resources**, **Hardware**, **Time**, and **Budget** being the sub-themes we identify. The codes in this theme answer the question of what the project allocated for testing and how.

> “ No additional resources were allocated to testing... There were no dedicated testing machines, no cloud-based tools, nothing of the sort. For our personal duties, we weren’t allotted a certain amount of developing and testing time - rather, we were expected to allocate our own time accordingly.

”

-G8-I (Indie)

“We invested in an external team for about 1 week of QA testing. This was mostly done to pass console certification. More would certainly have been nice.”

-G4-I (Indie)

Indies unsurprisingly report a lack of resources, with NO DEDICATED QA, TESTING DONE BY TEAM ON OWN TIME, and NO DEDICATED HARDWARE being the most mentioned items. Non-indies have DEDICATED QA STAFF OF AROUND 10 as the top mention followed by MULTIPLE (INTERNAL) QA TEAMS. Staffing mentioned by indies are on the order of 4 DEDICATED INTERNAL QA, 4–8 EXTERNAL PLAYTESTERS and BETA TESTING WITH AROUND 20 PLAYERS compared to non-indies' 100 INTERNAL QA TESTERS, EXTERNAL PLAYTESTING TEAMS OF UP TO 80 PEOPLE, and dedicated QA positions like QA MANAGER, QA DIRECTOR, and DEDICATED RELEASE QA MEMBER. Continuing the trend, more indies mention having no dedicated hardware for testing than the inverse, which forced them to use equipment they already owned. Meanwhile, non-indies either state that they always have the hardware they need or do not mention hardware at all. They mention processes for requisitioning hardware from their IT department and having access to build farms, implying that testing hardware is available and managed by others.

“The internal testing team was a small unit (under 10), while the external testing team size varied with the needs of the project (up to 80 at peak). Staffing needs were discussed between the QA manager and the leads of the external teams, and submitted to executive management for budget approval. We sometimes got pushback on the budget, but not often. Hardware was requested as needed from the IT department and our requests were always fulfilled.”

-G7-N (Non-Indie)

> We had and have hundreds of testers on this project and it didn't/doesn't feel like enough sometimes.
>
> *-G10-N (Non-Indie)*

Both groups mention team members being responsible for allocating part of their time to testing the game. Indies state that they avoid too much time investment on testing due to their small team, and that testing takes longer because of the size of their team. Non-indies talk about developers spending time to writing automated tests, which they find to be worthwhile in the long run. More importantly, a number of non-indies mention a TON OF OVERTIME, EXTREME OVERTIME FOR TESTING, and MANY QA WORKED OVERTIME NEARING RELEASE. Extreme overtime and being forced to work long hours, often without extra pay, is a known issue in the industry called "crunch". Crunch can lead to decreased mental health, burnout, and even divorce [26], [71].

Indies are the only ones to specify a testing budget with one project stating that they had $25,000 allocated for testing. Non-indies are less specific but mentioned MAJOR QA INVESTMENT, HEAVILY INVESTING IN ALL TESTING RESOURCES, and that they DID NOT GET MUCH PUSHBACK ON BUDGET REQUESTS. Indies are mixed in their sentiments about testing resources; one project feels that their resource constraints led to an inferior product whereas another feels that their internal and beta testers are adequate for their needs. In terms of bang for their buck, they PREFER ANOTHER DEVELOPER TO A DEDICATED QA and note that CLOUD DIAGNOSTICS WERE 1/50TH OF THE COST OF ANOTHER EMPLOYEE. On the other hand, non-indies with their massive resources feel that they still need more testers and QA.

4.5.7 Test Timing

Test timing concerns when tests are run, whether they are run on a regular schedule or conditionally when some criteria are met.

> I would test as I was adding new features or mechanics, then I would make a build when I felt there was enough new stuff to test. I would then send it out to my teammates and some friends to get feedback on.
>
> *-G11-I (Indie)*

> We would do testing on features and content as we worked. We would then send our feature/content to all other team members who would play and test the content... When all bugs from that round had been fixed, it would be sent out to team members again to see if they could reproduce the original bugs or find new ones. This would continue until no more bugs were found.
>
> *-G20-I (Indie)*

Indies describe more conditional tests than recurring tests, which is the opposite of non-indies. The only specifically recurring tests indies mention are SMOKE TESTS EVERY MORNING BY QA and WEEKLY PUBLIC BETA TESTS. Non-indies describe recurrences ranging from multiple times a day (HOURLY BUILDS OF ACCUMULATED CHANGES TO QA, VALIDATION TEST AT THE BEGINNING AND END OF EACH DAY) to numerous daily tests (DAILY SMOKE TESTS, multiple mentions of DAILY BUILD VERIFICATION TESTS) to weekly and monthly tests (EVERY FEATURE IN THE GAME TESTED WEEKLY, SMALL FOCUS TESTS DONE MONTHLY).

> Every morning, a small test ops group would complete a Build Verification Test on all systems and a cross section of content to ensure there were no blocking issues... as we approached submission to first party, we ran test passes against the certification requirements over the course of several weeks
>
> *-G12-N (Non-Indie)*

> We have a daily process where the QA team goes through the build and writes up tickets for issues that pop up while in gameplay. During this process the QA team goes through specific gameplay elements and content on a rotating schedule for total coverage.
>
> *-G14-N (Non-Indie)*

For conditional tests, non-indies' responses cover automated and manual checking of content before check-ins and merges as well as running smoke tests after bugfixes. Indies note multiple times that testing is done by developers while they are implementing features and list a number of conditional cases that describe a cascading chain of testing. For example, once developers test their implementation, it is handed off to others for additional testing (TESTING DONE BY TEAM AFTER DEV VERIFICATION, TESTING DONE BY QA AFTER DEV VERIFICATION) and passed on (EXTERNAL PLAYTESTING AFTER INTERNAL VALIDATION). They also discuss regression tests after bugfixes, integration and regression tests after new feature implementation, and build verification prior to releases.

These codes reinforce the finding that indies generally do not take a systematic and comprehensive approach to testing, preferring to test on-demand as situations arise. A reliance on chains of on-demand testing may lead to bottlenecks or breakdowns in process, whereas regular recurring testing would be better at ensuring that defects do not fall unnoticed through the cracks.

4.5.8 Test Automation

Automation codes address how was automated tools are used for testing on the projects described. We accept any level of automation in this category regardless of complexity as long as it performs some kind of testing and produces results.

> I haven't used many automation tools so I am curious about that now.
>
> *-G1-I (Indie)*

> It was pretty much all manual. I think I tried one automated tool but it didn't play well with the laptop I was using for development at the time. I also wasn't sure if taking the time to learn such tools would ultimately pay off with the limited resources.
>
> *-G16-I (Indie)*

The majority of indies indicate that they do NO AUTOMATED TESTING and an additional 3 mentions point to ALMOST EXCLUSIVELY MANUAL TESTING for indie projects. Automated tests used on indie projects include scripts for verifying and fixing scenes and data correctness, automated reporting and build verification, and automated build processes for merges to the main development branch. Non-indies mostly mention that they perform manual testing over automated, but many also mention that the majority of testing is automated on their project, indicating a more even distribution than indies and the overall importance and prevalence of manual testing even in AAA games. Non-indies use automation for common and tedious tasks, integration testing, and network testing. They also mention an AI-driven bot that runs 24/7 to trigger crashes and asserts. Automated tests are written by developers who implemented the respective feature, and one project points out that automated and manual tests have separate ownership.

> Majority manually, but automation saved us loads of time on things like invites to and from certain menus.
>
> *-G10-N (Non-Indie)*

> We had an extensive suite of debug tools, which is not quite test automation, but it was immensely helpful. In terms of automated testing we had an AI-driven bot system that would run 24/7 to increase test coverage and trigger asserts and crashes.
>
> *-G7-N (Non-Indie)*

The response to automated testing for indies is primarily negative or skeptical. They feel that automation is 4–5X MORE WORK [THAN MANUAL] DUE TO COMPLEX-

ITY OF FEATURES, automated testing for all merges is costly, and that investment to learn and implement automation may not pay off. The project that shipped automated tests with their game that automatically attached logs and screenshots to bug reports found it helpful, and the tool is able to catch errors that users may miss. Non-indies rely more on automated testing and find that it increases test coverage, but have difficulty maintaining automated tests in the face of pressure to implement new features. One respondent states that they HANDLED AUTOMATED TESTS WRONG on the project, stemming from misconceptions about automation and a lack of resources for them, which led to poor outcomes and an unmanageable backlog of tests.

This theme reveals that both groups perform the majority of their tests manually, but non-indies use more automation than indies. Automation is costly to implement and maintain, highlighting a need for well-defined goals and processes for automated tests. There are multiple mentions from both groups about a lack of awareness and understanding of automation in QA that may contribute to these difficulties.

4.5.9 Test Results

We gather codes relating to results and outputs from testing into this theme.

> “We used $debug.log()$ a lot.”
>
> *-G11-I (Indie)*

> “We had a group discord, when things came up they were put there. We liked to call it the “explain it like I’m 5” rule. We tried to simplify our problems so that just about everyone on our team could understand why what was doing what.”
>
> *-G17-I (Indie)*

Indies have informal or straightforward outputs from their tests from mentions of NO TESTING OUTPUT, TEXT-BASED LOGS, and VERBAL TEST OUTPUT. There are a

few well-defined indie outputs such as QA RESULTS AS FORMAL DOCUMENT TAGGED TO RELEASE for a team that has dedicated QA. For monitoring results, indies document bugs on Jira, Trello, and Discord, and one project mentions results being tracked by leads or extracted into charts and summaries. Another project has QA results documented using a form which they find to be helpful as their QA is led by non-technical members.

> “Our specific QA team had very strict adherence to the QA managers bug template. The manager also had a habit of "reprimanding" any tester who’d fall out of line with the template.”
>
> *-G6-N (Non-Indie)*

> “Automated tests would specify which tests failed. It would also specify the particular item in the test logic that failed. Regression and Smoke tests by the QA department would yield specific and reproducible test steps that developers could look at in order to fix bugs and defects.”
>
> *-G22-N (Non-Indie)*

Results on the non-indie side are more complex. They describe formal bug report templates that are strictly enforced by management, stating that deviations from the template are “discouraged” by management. They also describe specific output types, such as smoke and regression tests that include reproducible test steps and standardized outputs from CI. Non-indies also have more tracking and visualization for their output data, noting that they track all checks and have multiple dashboards for accessing and monitoring bugs, performance, telemetry, and build status.

Indies describe their test results primarily in terms of documentation, whereas non-indies include information about their purpose, such as reproducing failures, sending results to publishers, or visualizing the results as a proxy for measuring overall game health. Again, we find these results to be in line with our quantitative outcomes that non-indies are more likely to use their testing results to validate

internal targets and visualize them over time as metrics. Our Mann-Whitney U-test results from Table 4.4 also indicate that indies' test results are more subjective and open to interpretation, which aligns with their stated nonspecific testing goals, unclear test plans, and non-standardized test outputs.

4.5.10 Overall Sentiment

In this theme are codes relating to how the project and testing went overall that are not relevant to any of the specific sentiment subthemes.

> "Overall, I felt like we had a balanced approach that worked well. We accomplished a lot on a limited budget."
>
> *-G9-I (Indie)*

> "Many obvious bugs that could have been discovered and fixed were left for our first customers to run into - some of them rendering the game unplayable... the quality of the game wasn't up to par with other games of a similar scope and budget in my opinion."
>
> *-G8-I (Indie)*

Indies have mixed reactions to their overall testing experience. Some feel that their TESTING WENT SMOOTHLY, a BALANCED APPROACH ON A LIMITED BUDGET WORKED WELL, and their project is the BEST TESTING EXPERIENCE THEY HAD. Others feel that they WANTED MORE TESTING, NEEDED MORE EARLY USER TESTING, DID NOT HAVE TESTING IN PLACE EARLY ENOUGH, and as a result, their GAME SHIPPED AT LOWER QUALITY COMPARED TO GAMES OF SIMILAR SCOPES AND BUDGET. They point to a lack of testing organization during development, a general inexperience with testing, and a lack of resources as negative impacts. Some non-indies are very satisfied with their process, feeling that their TESTING PROCESS WAS VERY SUCCESSFUL and their FINAL PRODUCT WAS HIGH QUALITY WITH FEW ISSUES. Others describe their experience as DEVELOPMENT HELL with TONS OF BURNOUT and everyone laid off at the end of the project. These comments come from the projects that describe extreme overtime and a push to release the game without focusing on design.

> The final stages and release of the product had a very little issue and was a high-quality deliverable.
>
> *-G3-N (Non-Indie)*

> This project was brutal, and one of the best examples I've ever been in as far as "traditional" QA practices. There was a ton of burn out, a ton of overtime, and everyone got laid off at the end.
>
> *-G12-N (Non-Indie)*

Issues with testing for indies appear to stem from processes and resources, namely a lack thereof, whereas issues with testing for non-indies appear to stem from culture and management.

4.6 General Testing

At the end of our survey, we ask respondents three subjective questions related to testing. For coding purposes, we categorize these responses under **Overall Sentiment** but report them separately here due to their importance. We summarize the responses from these questions below, and also offer insight into answers concerning game development culture that respondents felt had significant impact on QA quality. As we do not ask questions pertaining to culture, we want to highlight these responses as participants felt that they were important points to raise.

4.6.1 Helpful for Testing

> Proper means of communication, being able to openly ask questions without "feeling dumb" [made testing better]. We were all equal and we were all in it together.
>
> *-G4-I (Indie)*

Indies highlight automated tests and dynamic game configuration as helpful for testing, most likely because they reduce the amount of time and effort re-

quired, and also mention good organization as a key factor. They also cite specific test plans, dedicated build testing time, and testing focused around specific milestones as facilitating the testing process. For studio culture, they feel that all team members being treated equally as well as fostering an environment of open communication and "no dumb questions" makes testing smoother.

> Knowing clear goals, understanding the testing process, standardized documentation.
>
> *-G22-N (Non-Indie)*

Non-indies also find automation and dynamic game configuration helpful for testing. They prefer to have standardized documentation, clear requirements and goals, and debug tools to make testing more efficient. They also mention direct access to producers and developers as beneficial, indicating a preference for embedded QA that works closely with feature teams. They want not to be micromanaged and to have better guidance for new hires.

4.6.2 Pain Points for Testing

> I think at first it was a lack of knowledge of how to properly test, what resources were available and learning as we went.
>
> *-G4-I (Indie)*

Indies feel the most pain from a lack of resources—small teams, time constraints, and hardware support. They also feel that they lack understanding of general testing procedures, resources, and root causes of bugs, which made testing more difficult for them. Additional issues they face include automated testing of complex systems as well as maintaining automated tests.

> But biggest pain points have to do with the treatment of QA as valued members of the games labour force. Our contracts made us easily replaceable, our input on any other forum but the bug database often dismissed.

”

-G12-N (Non-Indie)

Non-indies identify unclear testing goals and priorities as the biggest pains in their testing experience. They face difficulties with unclear or inaccurate test results and conflicting test priorities. Like indies, they have problems maintaining automated tests and feel that QA does not understand the state of software testing. They also cite a number of issues with QA being treated as unskilled labor, paid low wages, feeling easily replaceable, and having their input dismissed.

4.6.3 Wants for Testing

“ QA is game dev and a skilled dicipline. Anyone who says otherwise is wrong. VALUE YOUR QA AND PAY THEM BETTER. ”

-G17-I (Indie)

Indies have the most wants related to automated testing: better automated functional testing, automated performance testing that can be run on a range of hardware, more efficient automated builds, dedicated automation infrastructure, and more general knowledge about automation for testing. They also want more time for testing, tests baked into systems, and easier access to outside help.

“ Full time QA in games is primarily seen as unskilled work, leading to low wages and staff trying to use it as a springboard. ”

-G2-N (Non-Indie)

“ Valuing QA more, both in including it as part of development and in supporting QA staff. Pay QA on the same level as other employees. QA are developers as well. ”

-G10-N (Non-Indie)

Non-indies want both TESTING TO BE MORE LIKE TESTING IN TECH COMPANIES with BETTER ENGINEERING PRACTICES and GAMES TO BE LESS LIKE SOFTWARE. They

want more automated testing overall, less focus on unskilled QA, and more focus on scalable testing methodologies. Better treatment of QA is the most-mentioned item from non-indies, followed by better pay for QA, both of which are also mentioned by their indie counterparts. They additionally call for QA to be integrated into the full development process from design to implementation, better QA career progression, and QA to be recognized as game developers. By comparison, QA in general software development industry is fully integrated into the practice and recognized as critical, with no shortage of academic and industry research being presented at flagship conferences dedicated to testing such as the International Symposium on Software Testing and Analysis [2]. ZipRecruiter also shows the US-based salary median for video game testers and QA to be approximately $45,000–$61,000 USD [104], [107] compared to approximately $69,000–$72,000 USD for general software [105], [106]. We are not aware of widespread cultural issues in software testing, such as software testers not being considered developers or being laid off en masse upon completion of a project.

4.7 Summary

In this chapter, we presented the demographic information for respondents and project information for game responses. We discussed the rationale behind how we divided our dataset and presented quantitative and qualitative analyses for the data gathered from our game testing survey. For quantitative analyses, we calculated the differences in medians between the groups for each prompt, tested for statistically significant divergences of distributions of responses using Mann-Whitney U-tests, and reported on opt-out responses. Through the tests, we discovered 9 prompts that had significant differences concerning planning, goals, results, and resources. Through our qualitative analyses, we dug deeper into the open-ended text responses to identify common themes that illustrated the current practices, goals, and needs in testing. We found that indies struggled with resources, automation, and test planning and non-indies had systematic and thorough testing procedures that used more automation than indies. We also highlighted im-

portant issues brought up by both indie and non-indie respondents concerning studio culture and crunch. In the next chapter, we synthesize these results to discuss our findings and generate recommendations for improving quality assurance practices.

Chapter 5

Discussion of Results

In this chapter, we review our quantitative and qualitative results against our research questions. Based on these findings, we generate recommendations for developers, industry, and educators for how to support the indie game development sector.

5.1 Goals and Planning

From the quantitative results, indies had a probability of 81.3% to agree that they have no testing plans for the majority of their project and 82.6% to agree that they do not have validation plans for their features prior to implementation. They also had lower agreement that QA personnel set their testing plans and priorities, in line with indies generally not having dedicated QA members. In their open-ended text responses, we see descriptions of a chaotic, unstructured testing approach that is mainly concerned with experiential playtesting over thorough systematic verification. Their stated goals are also vague and lacking in detail compared to non-indies, who describe a regimented approach to testing for uncovering actionable items. Non-indies have a far clearer idea of what is being tested, who is responsible for testing it, when or how often it is tested, and what effect the outcomes of testing have.

Observations: Plans and Goals

- Indies have unclear goals for testing and lack testing plans overall

- Indie testing is unstructured and focused on player experience

We find the lack of well-defined testing goals and concrete test plans to be one of the biggest issues that indies report because it has downstream effects on the entire testing process. Although planning concretely requires more up-front time investment and may change frequently over the course of the project, having clear goals and plans makes it easier to select the best tests to achieve those goals, allocate resources accordingly, and obtain actionable results. Once goals and plans are set, they should be well-documented and available to the entire team so that everyone has a common understanding of what the team is working toward. Proper application of agile methodologies may also help with frequently changing goals and plans.

Recommendations: Plans and Goals

- Have clearly defined goals with actionable plans prior to testing
- Have good testing process documentation and ensure the team understands it

5.2 Testing Procedures and Timing

We have not found any statistically significant differences between the groups for performing tests. From the qualitative data, indies report performing fewer types of tests than non-indies in their open-ended text responses and do not discuss as many aspects of testing as non-indies do. Both groups highlight the importance of regression tests and question the effectiveness of unit testing in certain areas, such as UX testing. When it comes to actually running the tests, indies describe operating on a conditional basis, primarily performing tests when they find it necessary or expedient, as opposed to non-indies who describe testing systematically on a regular recurring schedule. Indies' conditional testing is also described as a chain of responsibility, with one step in the testing process triggering the next, which is fragile to bottlenecks and breakdowns should there be a failure along the chain.

Observations: Performance

- Regression tests are the most-mentioned tests from both indies and non-indies
- Indies perform more conditional testing; non-indies perform more regularly schedule testing
- Indies perform fewer types of tests than non-indies

Both groups mention regression testing numerous times, indicating that it is an important or frequently-run test, or both. Since it is a key test, indies should make the regression testing pipeline as easy and efficient as possible, such as by using automation. Since indies have fewer resources, they should spend the most time and care on tests by order of frequency and severity and choose the best-suited tests to achieve their goals. To do this, they must keep up-to-date on the state of testing practices and tools in the industry and understand their strengths and weaknesses. Since indies are already constrained for personnel and time, industry resources and studies on testing practices and tools with accessible summaries or workshops would benefit developers.

Recommendations: Performance

- Make regression testing easy and efficient
- Regularly schedule important tests and assign areas of responsibility
- Utilize a variety of testing practices for more effective and focused testing

5.3 Resources

The largest differences between indie and non-indie developers are the resources they have available. Our quantitative results reveal that testing for indie games is largely performed internally on the team and almost exclusively done by team members who have other primary tasks. 91.4% of indies are likely to say that they do not have the appropriate testing tools for their given project and goals than

non-indies, and 85.1% of indies are likely to say that they do not have enough people for testing on their project. These results are reinforced by our qualitative findings that indies lack dedicated QA personnel, perform testing on their own time on top of other duties, and do not have access to dedicated testing hardware. They feel that these factors led to less availability for testing, longer testing times, and inferior product outcomes compared to what they could accomplish with more resources. Non-indies by comparison have dedicated QA staff, access to external testers, time and budget allocated for testing, and extensive hardware availability. However, even though non-indies describe an abundance of time, personnel, money, and equipment for testing, many still report wanting more testing resources, which is likely connected to their reports of extreme overtime and crunch. Therefore, we suggest that access to resources is not as important as the proper fit of those resources to achieve the desired outcomes.

Observations: Resources

- Indies who perform testing have other primary roles
- Indies feel that they do not have appropriate testing tools
- Indies do not have enough testers or hardware for testing

Making the most of scarce resources requires indies to be thoughtful and innovative. As we see from non-indies, sheer numbers of testers are not the solution to testing issues—developers must be judicious in allocating resources wisely in accordance with testing goals and project priorities instead of throwing bodies at the problem. Again, having clearly defined goals and plans with actionable outcomes helps to determine how and when resources should be used. Indies should remain cognizant that costs that can not be met in terms of personnel, equipment, or funding are paid in time, which has a material cost to the project and team members, and beware the dangers of crunch that can result from repeatedly paying down technical debt with time.

Recommendations: Resources

- Be judicious in allocating resources to ensure they meet testing goals
- Be cautious of crunch; do not allocating excessive testing time to make up for shortfalls in planning or resources

5.4 Results and Outputs

Indies find the results from their testing to be subjective and open to interpretation with 83.1% of indies agreeing more strongly than non-indies. Comparatively, the majority of non-indies feel that their results are clear and objective. A focus on experiential playtesting, lack of clearly defined goals, and unstandardized documentation procedures factored into the subjectiveness of indie results. Indies also describe outputs that lack variety and specificity, such as text-based logs, verbal test output, and brief text notes about issues. Only 18.9% of indies agree more than non-indies that they track or visualize results over time as metrics. This is not surprising given that they are testing without clear, standardized outputs. Conversely, non-indies mention formal report templates, standardized output from tools, and access to dashboards for tracking various telemetry and game health indicators.

Observations: Results

- Indies' results are subjective and open to interpretation
- Indies' test outputs lack variety, specificity, and standardization
- Indies do not track or visualize their results over time as metrics

Indies should use testing results to validate targets and guide development priorities, which is difficult to do if those results are subjective and vague. We recommend that they create clear and standardized testing results and reports, and plan for how they will actionably address the results. Standardizing test results can also help make it easier for indies to track and visualize key metrics for the game over time. We recommend identifying simple key metrics that are measurable and maintainable to serve as proxies for the quality or health of the game. For

example, indies may not be able to gather and visualize detailed telemetry from their games, but tracking how many game crashes they saw in playtesting or mean time to players reaching a failure state in the game by build are simple metrics for game health and difficulty.

Recommendations: Results

- Create clear and standardized testing results and reports
- Plan for how to actionably address results before producing them
- Understand key metrics for the game; track and visualize them over time

5.5 Automation

Nearly all indies state that they perform very little to no automated testing on their project, whereas non-indies are more evenly split in their proportion of automated to manual testing but report more manual testing overall. Indies use automation for build processes and build verification, data and scene validation, and integrated tests that shipped with the game to make bug reporting easier. These tools are successful at achieving their aim but require some configuration for effective use. Indies who do not use automation approach it with skepticism; they find it to be several times more work than manual testing, which is costly given their already-constrained time resources; are not sure about how well it would work; and worry that the investment required to learn and build automation will not pay off in the long run. Non-indies use automation to take over for common and tedious tasks, increase test coverage, perform integration tests, and run network tests. Developers are responsible for writing automated tests for the features they implement which uses up time they could have spent on the game itself, but they feel that the time spent on automation was worth it in the end. Non-indies say that pressure from above to implement new features interferes with their ability to spend time on automation and that not understanding the role of automation and not dedicating enough resources to their maintenance leads to poor results.

Observations: Automation

- Indies do little to no automated testing
- Indies lack knowledge of automated testing and are skeptical of using it
- Non-indies use automation for specific key tasks as well as common and tedious tasks

Automation is not a silver bullet that will fix testing issues for indies, nor is it without its pitfalls and drawbacks. Indies should take a focused and directed application of automation techniques to key tasks to maximize their effectiveness. For example, automating the most frequently run, time-consuming, or tedious tests would offer better returns than automating a low-priority test that is easily done manually. Indies need more support to do this, both in learning about automated testing and having more usable tools that fit their workflow. In particular, as the indie respondents to our survey overwhelmingly used the Unity engine, robust automated testing tools for Unity and general open-source automation tools would benefit indie developers most. When adopting automated testing, indies must explicitly plan for writing and maintaining those tests.

Recommendations: Automation

- Focus automation to key tasks that will maximize effectiveness
- Make automated testing knowledge more public and accessible
- Plan for the time and development power required to maintain automated testing

5.6 Tools

We have not found any major quantitative differences in sentiment between indies and non-indies when it comes to the tools they use. However, the qualitative results once again shows stark differences: indies primarily use free tools, many of which are general-purpose tools not specific to games or testing. Conversely,

non-indies use a large number of paid tools specifically developed to aid with testing, planning, documentation, and tracking. The first-party tools that non-indies develop to help with testing are more complex in their functionality and specific to their goals, whereas indie first-party tools are more simple and broad in their use. However, the complexity of non-indie first-party tools leads to scenarios where they are improperly maintained over time and become ineffective as a result.

Observations: Tools

- Indies primarily use free/open-source tools which are not necessarily made for games or testing
- Improper maintenance of tools can lead to them becoming ineffective

Indies should select tools that will remove obstacles to development and help them achieve their goals. Similar to our recommendations for automation, this means addressing the most common, critical, repetitive, or tiresome tasks with proper tooling. Tools can also help enforce standardization, such as with test outputs, and help track or visualize data as well. Debugging tools and game state editors, for example, can help make testing much faster by setting up the environment conditions testers want to replicate with simple commands as opposed to going through a series of manual time-consuming steps or playing through irrelevant portions of the game to reach a certain spot. Similar to automated testing, tools must be updated along with the game in order for them to remain useful.

Recommendations: Tools

- Use tools to address the most common, critical, repetitive, or tiresome tasks
- Debug tools and game state editors can save time
- Allocate time to tools upkeep

5.7 Culture

Non-indies have the most concerns about culture and treatment of QA. They describe nightmare projects with an excess of overtime that lead to burnout, being laid off en masse upon project completion, and QA personnel who are treated as inferior to the rest of the development team. Indies do not report experiencing the same level of workplace toxicity, most likely due to QA generally being an added responsibility to existing primary roles instead of a dedicated position. Still, they are also affected, since, as respondents described in the qualitative answers, the poor treatment and opinion of QA work in the industry at large leads to QA feeling alienated, fewer people learning QA methodologies, and less interest in QA as a career option, all of which also negatively impacts indie development. The poor treatment of QA in non-indie games also harms indie developers by damaging the overall state of testing for games.

Observations: Culture

- QA are treated as inferior to the rest of the team or not considered game developers
- QA are frequently overworked and underpaid
- QA input on non-testing matters are often dismissed

Finally, culture is a major issue in the game industry that has been receiving more public focus recently, especially the exploitation of game workers and toxic development practices such as forced crunch. While the indies we surveyed generally do not have dedicated QA positions, they can still be part of the conversation for better working conditions and worker treatment in the industry. We believe that better treatment of QA—including recognizing them as game developers who have equal standing on the team, better pay, more career progression options, and integration into more of the development process—will lead to improved testing outcomes, higher quality games, and more ethical development practices.

Recommendations: Culture

- Treat QA as equal members of the team
- Pay QA better and give opportunities for career progression
- Integrate QA into the full development process from design to implementation

5.8 Research Questions

5.8.1 What are the most significant differences in game testing between indie and non-indie developers?

The most significant differences in game testing that we find between indie and non-indie developers are that indie developers have access to fewer resources, do not have clear testing goals and plans for features or the overall project, and do not use as much automated testing compared to non-indies. The majority of testing for indie games is done by team members who have other primary roles and not dedicated QA personnel, and indies feel that they need more testers and better tools for their project needs.

5.8.2 How much automated testing do indie developers use compared to non-indie developers, and what are the biggest pain points for automation?

Indie developers use almost exclusively manual testing with very little reliance on automation, whereas non-indies use slightly more manual than automated testing overall with some non-indie projects using automation for the majority of their tests. The biggest pain points for indies is a lack of knowledge of automated testing, a lack of accessible or compatible automation tools, and uncertainty about investing time into learning and creating automated tests. Non-indies face difficulties with contradicting priorities taking time away from automation, and warn against using automation without enough time or resources dedicated to maintaining them.

5.8.3 Where should indie developers focus their testing efforts and allocate their testing resources to maximize their return on investment?

We do not have enough detailed information in our responses to find answers to this research question. Instead, we address a similar question: *How can indie developers improve their approach to testing to maximize their return on investment?*

We recommend that indie developers first focus on defining clear testing goals. The more specific and defined the goals, the easier it is to test for them and obtain actionable results. Indies should keep current about testing methodologies and tools, including automation, and share their experiences and learnings with others. Without knowing the state of the practice, they cannot select the right tools for the job and may be missing out on more effective or efficient solutions. A key activity for increasing productivity is identifying the most critical tests and making them as frictionless as possible. Finding or creating tooling around these critical tests, such as automating the most frequently-run or time-consuming tests, frees up valuable developer time.

We also recommend that indies set a regular recurring schedule for testing instead of relying on ad-hoc practices. By making testing a routine activity, they can be more thorough and systematic about their testing tasks and coverage, and improve expectations and transparency around the QA process. Indies should also be careful to test in a way such that the results are not subjective and open to interpretation. Some subjectivity is impossible to avoid, especially in matters of user experience and player enjoyment, but results should generally be bounded and measurable to be actionable. Using mock tests or outputs to trial how those results will be used to effect change on the project can help illustrate misconceptions or gaps in the process. Objective results can also help them to track and visualize key metrics over the course of the project, which can be used to demonstrate development progress and game health.

Although these recommendations sound simple and straightforward to implement, indies' constraints make it difficult for them to allocate time to additional tasks such as researching the state of software testing practices or adding respon-

sibilities such as planning a testing schedule and evaluating it for coverage. While most people think of game development as being comprised of developers, designers, and creatives, there exists a large number of additional functions to be performed for a game studio. These functions include business planning, accounting, hiring, community management, training, advertising, networking, fundraising, project management, and more [58], [101]. While many non-indies have the capability to hire specialized personnel for these tasks, indies are forced to juggle them on top of their primary roles, leaving precious little time or energy for more work. The International Game Developers Association reports that 62% of game developers surveyed had a university or postgraduate degree, indicating that a sizeable proportion have not completed a degree. A lack of formal education may explain why some indies lack knowledge of software testing methodologies and development practices. We believe that offering indies our recommendations, the rationale for why these recommendations are important, and examples of how development and testing are done at other studios help to fill gaps in their knowledge and understanding.

5.9 Limitations and Threats to Validity

Our survey collected 22 game project entries from 19 respondents that were divided into two datasets of 11 games each. This sample size is relatively low, but is within acceptable boundaries for the Mann-Whitney U-tests and reflexive thematic analysis that we conducted. Our respondents also covered a broad range of industry experiences, with varying experiences in roles and project types. While we did not have much representation for developers with AA experience, the organization of our data into indie and non-indie groups obviated the need for a third category.

The respondents for our survey were self-selected from social media and online game development communities. They had a median of 6 years' experience in game development, and the majority of respondents were from Canada or between 25–34 years old. This may reflect our recruitment methods with younger demo-

graphics more likely to respond to online recruitment and Canadians more likely to see our posts. These findings therefore may not be generalizable to populations outside of this group.

Our survey was long with a minimum of 83 questions answered per respondent with a completion rate of 23% and an average completion time of 48 minutes for responses accepted into our final dataset. The length of our survey may have led to respondent fatigue, resulting in reduced attention and motivation in answering questions. We did not see evidence of a decrease in the quality of answers toward the latter end of the survey, such as straight-line answers on Likert-like responses or short, vague responses in open-ended text responses.

We used Likert-like prompts extensively in our survey. These instruments are susceptible to central tendency bias, or the likelihood of selecting more neutral choices than extremes, but we found most of our responses to be skewed toward one of the extremes. They can also show acquiescence bias, which is the tendency for respondents to simply agree with the prompt; we phrased our prompts in different ways to avoid this. For example, the two prompts "We primarily used testing tools we developed in-house" and "We primarily used third-party testing tools" would be in conflict if a respondent simply agreed with all prompts. We also randomized the order of rows for each set of Likert-like prompts.

For our project-based responses, we asked participants to answer based on their experiences on a completed commercial product. Respondents may be more likely to remember or report certain types of experiences than others, and their memory or sentiment regarding the process may have changed since that time.

5.10 Summary

We synthesized our quantitative and qualitative findings from Chapter 4 to offer recommendations for improving quality assurance for resource-constrained game developers in each major category of testing we identified. We returned to our research questions in light of our findings and recommendations. We talked about the most significant differences between and non-indie testing, differences in au-

tomation usage and major pain points, and suggestions for indie testing to maximize their resources. Finally, we wrapped up the chapter with a discussion about the limitations and potential threats to validity of this thesis.

Chapter 6

Conclusion

In this chapter, we give an overview of the thesis, summarize our contributions, and examine potential future work arising from this research.

The contributions of this thesis are a review of the state of testing in the game industry, an online survey that we deployed to learn about quality assurance practices, goals, and needs for games, and the findings and recommendations from our survey for improving testing for indie games. We were motivated by the growth and popularity of the video game industry and its importance to entertainment, culture, and art to examine the state of testing for games as a method of improving game quality and decreasing development costs. We focused on commercial indie game development in particular, because indie games represent the majority of games released each year. However, indie developers are constrained by their limited resources in comparison to large studios releasing blockbuster games. In Chapter 2, we presented an overview of the state of the industry, common testing terms and practices, and a literature review of game development, testing methodologies, bugs in games, and automated testing.

To find out about game developers' practices, goals, and needs for quality assurance, and to compare how indies and non-indies performed testing for their games, we created an online survey. We discussed how we piloted this survey in Chapter **??**; we explained the rationale behind its organization and questions, the pilot participants we recruited, and the feedback we received from them. We used that feedback to improve our survey before deploying it online and recruiting participants. In Chapter 3, we detailed the changes we made, how we recruited partic-

ipants, and our methodology for performing quantitative and qualitative analyses on the data we received. The major analyses we used were Mann-Whitney U-tests for Likert-like responses and reflexive thematic analysis for open-ended answers.

We presented our results in Chapter 4. The respondent demographics showed a variety of experiences spread out across various positions, project types, and skills, and the projects described had an even split of indie and non-indie games. We found 14 differences in medians that indicated a shift in overall opinion between the groups and conducted Mann-Whitney U-tests to discover which prompts had statistically significant differences in the distributions of their responses. We reviewed the nine prompts identified by the significance tests and reported their common language effect size. These findings showed that indies had less clarity in goals and planning, had less resources than they felt they needed, and did not use test results to validate internal targets or visualize key metrics over time. Our qualitative analysis gave additional context to the quantitative results and confirmed that indies had a lack of goals and planning, a primarily ad-hoc and exploratory approach to testing, and an overall lack of suitable tools and automation. We also found that respondents were frustrated with crunch and the treatment of QA in the game industry and discussed the culture and management issues they raised.

In Chapter 5, we combined our results from the quantitative and qualitative analyses to present our key findings and recommendations for goals and planning, testing procedures and timing, resources, results and outputs, automation, tools, and culture. Our recommendations for resource-constrained developers included having more clearly defined goals and actionable plans, testing on a regular schedule for important tests, creating standardized test results and reports, and addressing the most common, critical, repetitive, or tedious tasks using automation and tools. Finally, we returned to our research questions and highlighted from the responses the most significant differences in game testing between the groups, the differences and pain points for automated testing specifically, and how indie developers can improve their testing approach and procedures.

We see this work as being an initial exploration into the differences between indie and non-indie game development and testing with ample avenues for fur-

ther research. Replicating this study with more participants would be beneficial for confirming our findings and avoiding undue effects from a small sample size. We also see value in performing a similar study as a series of semi-structured interviews, as opposed to open-ended text responses on an online survey, to gain richer and more specific information from respondents and to clarify terminology that may have conflicting meanings. Due to the format of our survey, we were unable to follow up with respondents or dig deeper into salient points that were raised. We also think it would be worthwhile to compare practices between different indie studios, instead of comparing them against non-indies, due to the extreme variance that can exist within indies and the sometimes-massive gap between indie and AAA studios.

In conclusion, this thesis illustrates the need for more focused research in game development quality assurance practices. Indie developers' goals and processes for testing differ from non-indies in critical ways, and they do not have access to the same methods and tools to achieve similar goals. We believe that improving indie testing knowledge and practices (along with improving the treatment of QA) will lead to better games for less development cost, empower indie creators in their craft, and raise the standard of quality for the game industry overall.

References

[1] *2021 essential facts about the video game industry*, Feb. 2022. [Online]. Available: https://www.theesa.com/resource/2021-essential-facts-about-the-video-game-industry/.

[2] *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. [Online]. Available: https://conf.researchr.org/home/issta-2022.

[3] A. M. Albaghajati and M. A. K. Ahmed, "Video game automated testing approaches: An assessment framework," *IEEE Transactions on Games*, 2020, ISSN: 24751510. DOI: 10.1109/TG.2020.3032796.

[4] S. Ariyurek, A. Betin-Can, and E. Surer, "Enhancing the Monte Carlo tree search algorithm for video game testing," vol. 2020-August, 2020. DOI: 10.1109/CoG47356.2020.9231670.

[5] ——, "Automated video game testing using synthetic and humanlike agents," *IEEE Transactions on Games*, vol. 13, 1 2021, ISSN: 24751510. DOI: 10.1109/TG.2019.2947597.

[6] Atlassian, *Jira: Issue and project tracking software*. [Online]. Available: https://www.atlassian.com/software/jira.

[7] *Azure devops services: Microsoft Azure (software)*. [Online]. Available: https://azure.microsoft.com/en-us/services/devops/.

[8] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslen, "Augmenting automated game testing with deep reinforcement learning," vol. 2020-August, 2020. DOI: 10.1109/CoG47356.2020.9231552.

[9] R. Bergmann, J. Ludbrook, and W. P. J. M. Spooren, "Different outcomes of the Wilcoxon-Mann-Whitney test from different statistics packages," *The American Statistician*, vol. 54, no. 1, pp. 72–77, 2000, ISSN: 00031305. [Online]. Available: http://www.jstor.org/stable/2685616 (visited on 04/09/2022).

[10] I. Borovikov, Y. Zhao, A. Beirami, J. Harder, J. Kolen, J. Pestrak, J. Pinto, R. Pourabolghasem, H. Chaput, M. Sardari, L. Lin, N. Aghdaie, and K. Zaman, "Winning isn't everything: Training agents to playtest modern games," *AAAI Workshop on Reinforcement Learning in Games*, 2019.

[11] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006. DOI: 10.1191/1478088706qp063oa. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa.

[12] ——, "Reflecting on reflexive thematic analysis," *Qualitative Research in Sport, Exercise and Health*, vol. 11, no. 4, pp. 589–597, 2019. DOI: 10.1080/2159676X.2019.1628806. [Online]. Available: https://doi.org/10.1080/2159676X.2019.1628806.

[13] D. Byrne, "A worked example of Braun and Clarke's approach to reflexive thematic analysis," *Quality and Quantity*, 2021, ISSN: 15737845. DOI: 10.1007/s11135-021-01182-y.

[14] J. H. Bécares, L. C. Valero, and P. P. G. Martín, "An approach to automated videogame beta testing," *Entertainment Computing*, vol. 18, 2017, ISSN: 18759521. DOI: 10.1016/j.entcom.2016.08.002.

[15] B. Chan, J. Denzinger, D. Gates, K. Loose, and J. Buchanan, "Evolutionary behavior testing of commercial computer games," vol. 1, 2004. DOI: 10.1109/cec.2004.1330847.

[16] C. S. Cho, D. C. Lee, K. M. Sohn, C. J. Park, and J. H. Kang, "Scenario-based approach for blackbox load testing of online game servers," 2010. DOI: 10.1109/CyberC.2010.54.

[17] C. S. Cho, K. M. Sohn, C. J. Park, and J. H. Kang, "Online game testing using scenario-based control of massive virtual users," vol. 2, 2010.

[18] J. J. Cho, *Game testing survey artifacts*. [Online]. Available: https://github.com/gojeffcho/game-testing-survey-artifacts.

[19] J. Clement, *Video game market value worldwide 2020 to 2025*, Nov. 2021. [Online]. Available: https://www.statista.com/statistics/292056/video-game-market-value-worldwide/.

[20] R. Conroy, "What hypotheses do "nonparametric" two-group tests actually test?" *Stata Journal*, vol. 12, pp. 182–190, Jun. 2012. DOI: 10.1177/1536867X1201200202.

[21] M. Dawson, D. N. Burrell, E. Rahim, and S. Brewster, "Integrating software assurance into the software development life cycle (SDLC) meeting Department of Defense (DoD) demands," *Journal of Information Systems Technology and Planning*, vol. 3, 6 2010, ISSN: 19455240.

[22] S. Devlin, A. Anspoka, N. Sephton, P. I. Cowling, and J. Rollason, "Combining gameplay data with Monte Carlo tree search to emulate human play," *Proceedings of The Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)*, 2016.

[23] *Devsuite, application lifecycle software*. [Online]. Available: https://techexcel.com/products/devsuite/.

[24] *Devtrack, best project task management software.* [Online]. Available: https://techexcel.com/products/devtrack/.

[25] *Discord (app).* [Online]. Available: https://discordapp.com/.

[26] N. Dyer-Witheford and G. S. D. Peuter, ""EA Spouse" and the crisis of video game labour: Enjoyment, exclusion, exploitation, and exodus," *Canadian Journal of Communication*, vol. 31, 3 2006, ISSN: 0705-3657. DOI: 10.22230/cjc.2006v31n3a1771.

[27] R. Ebert, *Video games can never be art.* [Online]. Available: https://www.rogerebert.com/roger-ebert/video-games-can-never-be-art.

[28] *Frostbite - Electronic Arts Inc. (software).* [Online]. Available: https://www.ea.com/frostbite/engine.

[29] *Games industry data and analysis.* [Online]. Available: https://vginsights.com/insights/article/indie-games-make-up-40-of-all-units-sold-on-steam.

[30] P. García-Sánchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo, "Automated playtesting in collectible card games using evolutionary algorithms: A case study in Hearthstone," *Knowledge-Based Systems*, vol. 153, 2018, ISSN: 09507051. DOI: 10.1016/j.knosys.2018.04.030.

[31] *Git Large File Storage (software).* [Online]. Available: https://git-lfs.github.com/.

[32] S. F. Gudmundsson, P. Eisen, E. Poromaa, A. Nodet, S. Purmonen, B. Kozakowski, R. Meurling, and L. Cao, "Human-like playtesting with deep learning," vol. 2018-August, 2018. DOI: 10.1109/CIG.2018.8490442.

[33] A. Hoyt, M. Guzdial, Y. Kumar, G. Smith, and M. O. Riedl, "Integrating automated play in level co-creation," 2019. DOI: 10.48550/ARXIV.1911.09219. [Online]. Available: https://arxiv.org/abs/1911.09219.

[34] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," 2015. DOI: 10.1109/MODELS.2015.7338274.

[35] A. Jaffe, A. Miller, E. Andersen, E. A. Y. E. Liu, A. Karlin, and Z. Popović, "Evaluating competitive game balance with restricted play," 2012.

[36] *Journey.* [Online]. Available: https://store.steampowered.com/app/638230/Journey/.

[37] *Journey (2012 video game)*, Mar. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Journey_(2012_video_game).

[38] Y. W. Jung, B. H. Lim, K. H. Sim, H. J. Lee, I. K. Park, J. Y. Chung, and J. Lee, "VENUS: The online game simulator using massively virtual clients," vol. 3398, 2005. DOI: 10.1007/978-3-540-30585-9_66.

[39] J. Kasurinen, "Games as software - similarities and differences between the implementation projects," vol. 1164, 2016. DOI: 10.1145/2983468.2983501.

[40] J. Kasurinen and K. Smolander, "What do game developers test in their products?," 2014. DOI: 10.1145/2652524.2652525.

[41] O. Keehl and A. M. Smith, "Monster Carlo: An MCTS-based framework for machine playtesting unity games," vol. 2018-August, 2018. DOI: 10.1109/CIG.2018.8490363.

[42] F. Laricchia, *Video games industry revenues 2000-2014*, Jun. 2010. [Online]. Available: https://www.statista.com/statistics/268954/revenues-of-global-video-games-industry/.

[43] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What went wrong: A taxonomy of video game bugs," 2010. DOI: 10.1145/1822348.1822363.

[44] C. Love, *Digital: A Love Story*. [Online]. Available: https://scoutshonour.com/digital/.

[45] ——, *What kind of year has it been?* Jan. 2011. [Online]. Available: https://blog.loveconquersallgames.com/post/2605257258/what-kind-of-year-has-it-been.

[46] A. Mak, *Why testing video games all day for a living is actually pretty brutal*, Feb. 2022. [Online]. Available: https://slate.com/technology/2022/02/activision-raven-union-video-games-testers-qa.html.

[47] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, vol. 18, 1 1947, ISSN: 0003-4851. DOI: 10.1214/aoms/1177730491.

[48] B. B. Marklund, H. Engström, M. Hellkvist, and P. Backlund, "What empirically based research tells us about game development," *The Computer Games Journal*, vol. 8, 3-4 2019. DOI: 10.1007/s40869-019-00085-1.

[49] K. O. McGraw and S. P. Wong, "A common language effect size statistic," *Psychological Bulletin*, vol. 111, 2 1992, ISSN: 00332909. DOI: 10.1037/0033-2909.111.2.361.

[50] T. McKenzie, M. M. Trujillo, and S. Hoermann, "Software engineering practices and methods in the game development industry," 2019. DOI: 10.1145/3341215.3354647.

[51] F. de Mesentier Silva, I. Borovikov, J. Kolen, N. Aghdaie, and K. Zaman, "Exploring gameplay with AI agents," 2018.

[52] L. Mugrai, F. Silva, C. Holmgard, and J. Togelius, "Automated playtesting of matching tile games," vol. 2019-August, 2019. DOI: 10.1109/CIG.2019.8848057.

[53] E. Murphy-Hill, T. Zimmermann, and N. Nagappan, "Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?," 2014. DOI: 10.1145/2568225.2568226.

[54] J. Musil, A. Musil, D. Winkler, and S. Biffl, "A survey on a state of the practice in video game development," Tech. Rep., Mar. 2010.

[55] N. Napolitano, *Testing match-3 video games with deep reinforcement learning*, 2020. DOI: 10.48550/ARXIV.2007.01137. [Online]. Available: https://arxiv.org/abs/2007.01137.

[56] *Nunit (software)*. [Online]. Available: https://nunit.org/.

[57] M. Ostrowski and S. Aroudj, "Automated regression testing within video game development," *GSTF Journal on Computing (JoC)*, vol. 3, 2 2013. DOI: 10.7603/s40601-013-0010-4.

[58] L. S. Pereira and M. M. S. Bernardes, "Aspects of independent game production: An exploratory study," *Comput. Entertain.*, vol. 16, no. 4, 2018. DOI: 10.1145/3276322. [Online]. Available: https://doi-org.login.ezproxy.library.ualberta.ca/10.1145/3276322.

[59] J. Pfau, A. Liapis, G. Volkmar, G. N. Yannakakis, and R. Malaka, "Dungeons replicants: Automated game balancing via deep player behavior modeling," vol. 2020-August, 2020. DOI: 10.1109/CoG47356.2020.9231958.

[60] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated game testing with ICARUS: Intelligent completion of adventure riddles via unsupervised solving," 2017. DOI: 10.1145/3130859.3131439.

[61] C. Politowski, L. Fontoura, F. Petrillo, and Y. G. Guéhéneuc, "Are the old days gone? A survey on actual software engineering processes in video game industry," vol. 16-May-2016, 2016. DOI: 10.1145/2896958.2896960.

[62] C. Politowski, Y.-G. Guéhéneuc, and F. Petrillo, "Towards automated video game testing: Still a long way to go," Feb. 2022.

[63] C. Politowski, F. Petrillo, and Y. G. Guéhéneuc, "A survey of video game testing," 2021. DOI: 10.1109/AST52587.2021.00018.

[64] C. Politowski, F. Petrillo, G. C. Ullmann, and Y. G. Guéhéneuc, "Game industry problems: An extensive analysis of the gray literature," *Information and Software Technology*, vol. 134, 2021, ISSN: 09505849. DOI: 10.1016/j.infsof.2021.106538.

[65] *Redmine (software)*. [Online]. Available: https://www.redmine.org/.

[66] C. Reuter, S. Göbel, and R. Steinmetz, *Detecting structural errors in scene-based multiplayer games using automatically generated petri nets*, 2015.

[67] A. Saini, *Cost to fix bugs and defects during each phase of the sdlc*, Oct. 2021. [Online]. Available: https://www.synopsys.com/blogs/software-security/cost-to-fix-bugs-during-each-sdlc-phase/.

[68] C. Salge, C. Lipski, T. Mahlmann, and B. Mathiak, "Using genetically optimized artificial intelligence to improve gameplaying fun for strategical games," 2008. DOI: 10.1145/1401843.1401845.

[69] R. E. Santos, C. V. Magalhes, L. F. Capretz, J. S. Correia-Neto, F. Q. D. Silva, and A. Saher, "Computer games are serious business and so is their quality: Particularities of software testing in game development from the perspective of practitioners," 2018. DOI: 10.1145/3239235.3268923.

[70] C. Schaefer, H. Do, and B. M. Slator, "Crushinator: A framework towards game-independent testing," 2013. DOI: 10.1109/ASE.2013.6693143.

[71] J. Schreier, *The horrible world of video game crunch*, Sep. 2016. [Online]. Available: https://kotaku.com/crunch-time-why-game-developers-work-such-insane-hours-1704744577.

[72] ——, *Quality assured: What it's really like to test games for a living*, Jan. 2017. [Online]. Available: https://kotaku.com/quality-assured-what-it-s-really-like-to-play-games-fo-1720053842.

[73] ——, *Former PlayStation chief muses on the future of gaming*, 2021. [Online]. Available: https://www.bloomberg.com/news/newsletters/2021-09-03/ex-playstation-chief-mulls-future-of-gaming-and-his-new-job.

[74] SciPy, *Scipy.stats.mannwhitneyu*. [Online]. Available: https://docs.scipy.org/doc/scipy-1.8.0/html-scipyorg/reference/generated/scipy.stats.mannwhitneyu.html.

[75] Y. Shin, J. Kim, K. Jin, and Y. B. Kim, "Playtesting in match 3 game using strategic plays via reinforcement learning," *IEEE Access*, vol. 8, 2020, ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2980380.

[76] F. D. M. Silva, S. Lee, J. Togelius, and A. Nealen, "AI as evaluator: Search driven playtesting of modern board games," vol. WS-17-01 - WS-17-15, 2017.

[77] ——, "AI-based playtesting of contemporary board games," vol. Part F130151, 2017. DOI: 10.1145/3102071.3102105.

[78] O. Solon, *MoMA to exhibit videogames, from Pong to Minecraft*, Nov. 2012. [Online]. Available: https://www.wired.com/2012/11/moma-videogames/.

[79] F. Southey, G. Xiao, R. Holte, M. Trommelen, and J. Buchanan, "Semi-automated gameplay analysis by machine learning," *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 1, no. 1, pp. 123–128, Sep. 2021. [Online]. Available: https://ojs.aaai.org/index.php/AIIDE/article/view/18727.

[80] F. Southey, G. Xiao, R. C. Holte, M. Trommelen, and J. Buchanan, "Semi-automated gameplay analysis by machine learning," 2005.

[81] S. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial playfulness: A tool for automated agent-based playtesting," 2019. DOI: 10.1145/3290607.3313039.

[82] *Steam store*. [Online]. Available: https://store.steampowered.com/.

[83] A. Styhre and B. Remneland-Wikhamn, "The video game as agencement and the image of new gaming experiences: The work of indie video game developers," *Culture and Organization*, vol. 27, 6 2021, ISSN: 14772760. DOI: 10.1080/14759551.2021.1919893.

[84] Superannuation, *How much does it cost to make a big video game?* Jan. 2014. [Online]. Available: https://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649.

[85] T. G. Tan, Y. N. Yong, K. O. Chin, J. Teo, and R. Alfred, "Automated evaluation for ai controllers in tower defense game using genetic algorithm," vol. 378 CCIS, 2013. DOI: 10.1007/978-3-642-40567-9_12.

[86] *TeamCity: The hassle-free CI and CD server by jetbrains*. [Online]. Available: https://www.jetbrains.com/teamcity/.

[87] *Test Rail: Test management & QA software for agile teams*. [Online]. Available: https://www.gurock.com/testrail/.

[88] *The art of video games*. [Online]. Available: https://americanart.si.edu/exhibitions/games.

[89] *The world's most popular free online survey tool*. [Online]. Available: https://www.surveymonkey.com/.

[90] *Trello (software)*. [Online]. Available: https://trello.com/.

[91] A. Truelove, E. S. D. Almeida, and I. Ahmed, "We'll fix it in post: What do bug fixes in video game update notes tell us?," 2021. DOI: 10.1109/ICSE-Companion52605.2021.00120.

[92] *Twitch (website)*. [Online]. Available: https://www.twitch.tv/.

[93] Unity Technologies, *Unity cloud build*. [Online]. Available: https://unity.com/features/cloud-build.

[94] ——, *Unity cloud diagnostics*. [Online]. Available: https://unity.com/products/cloud-diagnostics.

[95] ——, *Unity engine (software)*. [Online]. Available: https://unity.com/.

[96] ——, *Unity profiler (documentation)*. [Online]. Available: https://docs.unity3d.com/Manual/Profiler.html.

[97] *Unreal engine: The most powerful real-time 3d creation tool (software)*. [Online]. Available: https://www.unrealengine.com/.

[98] R. Vallat, *Pingouin.mwu*. [Online]. Available: https://pingouin-stats.org/generated/pingouin.mwu.html#pingouin.mwu.

[99] J. Walker, *Dev reveals exploitative nature of most game contracts*, Aug. 2021. [Online]. Available: https://kotaku.com/dev-reveals-exploitative-nature-of-most-game-contracts-1847497090.

[100] *What is an indie game and why should you care?* Apr. 2020. [Online]. Available: https://www.indiegamemag.com/what-is-an-indie-game/.

[101] J. R. Whitson, B. Simon, and F. Parker, "The missing producer: Rethinking indie cultural production in terms of entrepreneurship, relational labour, and sustainability," *European Journal of Cultural Studies*, vol. 24, no. 2, pp. 606–627, 2021. DOI: 10.1177/1367549418810082. [Online]. Available: https://doi.org/10.1177/1367549418810082.

[102] W. Witkowski, *Videogames are a bigger industry than movies and North American sports combined, thanks to the pandemic*, Dec. 2020. [Online]. Available: https://www.marketwatch.com/story/videogames-are-a-bigger-industry-than-sports-and-movies-combined-thanks-to-the-pandemic-11608654990/.

[103] A. Yessad, I. Mounier, J. M. Labat, F. Kordon, and T. Carron, "Have you found the error? A formal framework for learning game verification," vol. 8719 LNCS, 2014. DOI: 10.1007/978-3-319-11200-8_45.

[104] ZipRecruiter, *Quality assurance game tester salary*. [Online]. Available: https://www.ziprecruiter.com/Salaries/Quality-Assurance-Game-Tester-Salary.

[105] ——, *Software quality assurance analyst annual salary*. [Online]. Available: https://www.ziprecruiter.com/Salaries/Software-Quality-Assurance-Analyst-Salary.

[106] ——, *Software tester annual salary*. [Online]. Available: https://www.ziprecruiter.com/Salaries/Software-Tester-Salary.

[107] ——, *Video game quality assurance salary*. [Online]. Available: https://www.ziprecruiter.com/Salaries/Video-Game-Quality-Assurance-Salary.

[108] A. Zook, E. Fruchter, and M. O. Riedl, *Automatic playtesting for game parameter tuning via active learning*, 2019. DOI: 10.48550/ARXIV.1908.01417. [Online]. Available: https://arxiv.org/abs/1908.01417.

[109] A. Zook, B. Harrison, and M. O. Riedl, *Monte-Carlo tree search for simulation-based strategy analysis*, 2019. DOI: 10.48550/ARXIV.1908.01423. [Online]. Available: https://arxiv.org/abs/1908.01423.

Appendix A

Pilot Survey Questions

A.1 Respondent Information

A.1.1 Information and Consent

Participants were shown the information, contact, and consent sheet for this survey along with a link to download a PDF copy for their records.

(PQ1) *Do you consent to participate in this survey?* Yes/No

A.1.2 Eligibility

(PQ2) *Do you have at least two years' experience in commercial game development in the past 10 years?* Yes/No

(PQ3) *Have you worked for at least one year on a game that has been commercially released?* Yes/No

A.1.3 Demographics

(PQ4) *How many years have you been active in the games industry?* Decimal text input between 0–100

(PQ5) *Which of the following roles have you previously held in the games industry?* Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner

(PQ6) *How many years of indie game development experience do you have, in the past 10 years?* Integer input between 0–10

(PQ7) *How many years of AA game development experience do you have, in the past 10 years?* Integer input between 0–10

(PQ8) *How many years of AAA game development experience do you have, in the past 10 years?* Integer input between 0–10

A.2 Project-Based Responses

Project Information

(PQ9) *Please enter a nickname for the project you have chosen. This is used purely for reference purposes and does not have to be the actual name of the game.* Text response

(PQ10) *How would you classify the project?* Indie Game, AA Game, AAA Game

(PQ11) *What was your primary role on the project?* Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner

(PQ12) *What were your other roles on the project, if any?* Multi-Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner

(PQ13) *What engine did you use for the project?* Text response

(PQ14) *How long were you working on this project?* 1 year or less; 2-3 years; Over 3 years

(PQ15) *What was the size of the team for this project?* 1–49; 50–99; 100–499; 500 or more

(PQ16) *What was the approximate budget for this project?* Unknown; Under $1M USD; $1M–$10M USD; Over $10M USD

(PQ17) *Did you have a publisher for this project?* Yes, we had a publisher who provided marketing support; Yes, we had a publisher, but they did not provide marketing support; No, we did not have a publisher

(PQ18) *How long was this project in development?* 1 year or less; 2–3 years; Over 3 years

Test Performance

(PQ19) *Please describe how testing was performed on [this project]. What steps were taken by testing, by whom, and when?* Open-ended text response

Respondents rated their agreement with the following prompts on a five-point Likert-like scale: Strongly Disagree, Somewhat Disagree, Neither Agree Nor Disagree, Somewhat Agree, Strongly Agree. They could also select an "N/A" option if the prompt was not applicable to their experience.

(PQ20-a) *We primarily used testing tools that we developed in-house*

(PQ20-b) *We primarily used third-party testing tools*

(PQ20-c) *Team members were able to give actionable input and feedback to the testing process*

(PQ20-d) *The majority of testing was done by a dedicated quality assurance team*

(PQ20-e) *We routinely ran unit tests on this project*

(PQ20-f) *We routinely ran regression tests on this project*

(PQ20-g) *We routinely ran integration tests on this project*

(PQ20-h) *We routinely ran smoke tests on this project*

Test Planning

(PQ21) *Please describe the test planning process for [this project]. How were test plans made, by whom, and when?* Open-ended text response

Likert-like agreement rating prompts:

(PQ22-a) *The testing process was well documented and openly available to the team*

(PQ22-b) *The testing process followed the testing plan in most cases without requiring ad-hoc changes*

(PQ22-c) *At any given point during development, I had a good idea of the current testing plan including tools, priorities, and schedule*

(PQ22-d) *Each major feature for development had a testing or validation plan prior to implementation*

(PQ22-e) *There were no concrete testing plans or priorities for the majority of this project*

(PQ22-f) *Test plans and priorities were primarily set by developers*

(PQ22-g) *Test plans and priorities were primarily set by quality assurance personnel*

(PQ22-h) *Test plans and priorities were primary set by supervisors or leads*

(PQ22-i) *Test plans or priorities were primarily set by the producer, game director, or senior management*

Testing Goals

(PQ23) *Please describe the goals of testing that you and your team had on [this project]. What were you testing for, and how did your testing process accomplish that goal (or fall short)?* Open-ended text response

Likert-like agreement rating prompts:

(PQ24-a) *We routinely performed testing for game mechanics (e.g. game rules, balance between features, overall content design)*

(PQ24-b) *We routinely performed testing for technical aspects of the game (e.g. functionality, stability)*

(PQ24-c) *We routinely performed testing for user experience (e.g. fun factor, user impression, satisfaction)*

(PQ24-d) *The results of testing were often subjective and open to interpretation*

Test Automation

(PQ25) *Please describe the degree of manual and automated testing that you experienced on [this project], including whether there were specific areas that were prioritized for testing manually or with automation.*

Likert-like agreement rating prompts:

(PQ26-a) *We used more manual testing than automated testing*

(PQ26-b) *We had personnel dedicated to writing scripted or automated tests on this project*

(PQ26-c) *The automated testing tools that we used required frequent adjustments as the project progressed*

(PQ26-d) *Manual testing yielded more actionable results than automated testing*

(PQ26-e) *Automated testing yielded more actionable results than manual testing*

(PQ26-f) *Manual testing yielded results that could not be determined through automated testing*

(PQ26-g) *Automated testing yielded results that could not be determined through manual testing*

(PQ26-h) *Automated testing is important for game development*

(PQ26-i) *I am happy with the proportion of automated testing to manual testing that was done for this project*

(PQ26-j) *Automated testing was a source of frustration on this project*

Testing Tools

(PQ27) *Please list the tools you used for testing on [this project], whether they were used in a certain order, as well as what you perceived as the major benefits and drawbacks to each one (with consideration for functionality and usability).* Open-ended text response

Likert-like agreement rating prompts:

(PQ28-a) *The tools that we used for testing were developed specifically for games*

(PQ28-b) *The tools that we used for testing were intuitive and easy to use*

(PQ28-c) *The tools that we used for testing were well-suited to our testing goals*

(PQ28-d) *The output from testing was easy to read and understand*

(PQ28-e) *The tools that we used for testing complemented each other well*

(PQ28-f) *There was a structured approach to testing that was repeated throughout the project*

Test Results

(PQ29) *Please describe the format, readability, and usability of the testing output, and how the testing results and output were used by the [project] team.* Open-ended text response

Likert-like agreement rating prompts:

(PQ30-a) *The results of testing were used to form actionable plans to address the results*

(PQ30-b) *The results of testing could cause major changes to the game's design*

(PQ30-c) *The results of testing were used to validate internal targets*

(PQ30-d) *The results of testing were primarily used by the testing or development team*

(PQ30-e) *The results of testing were used primarily by leads and managers*

(PQ30-f) *The results of testing were monitored or visualized over time and used as metrics*

Testing Resources

(PQ31) *Please describe your understanding of the allocation of resources for testing on [this project]. What resources were allocated to testing (human resources, funding, time allotment, hardware, etc.) and how?* Open-ended text answer

Likert-like agreement rating prompts:

(PQ32-a) *We had sufficient testing resources overall to support this project*

(PQ32-b) *We had sufficient testing/QA personnel dedicated to this project*

(PQ32-c) *We had sufficient test automation for this project*

(PQ32-d) *We had appropriate testing tools given the project and its testing goals*

(PQ32-e) *The team requested additional testing resources during the project*

(PQ32-f) *Testing was primarily contracted to an outside firm*

(PQ32-g) *Testing was primarily performed internally on the team*

(PQ32-h) *Testing was done primarily by team members who had other primary tasks, such as developer or designer*

(PQ32-i) *Testers were integrated with the development team and had regular direct access to developers*

Additional Information

(PQ33) *Would you like to add any additional details about [this project]? Optional open-ended text response*

(PQ34) *Would you like to discuss another game project that you worked on for at least one year? If you select "yes", you will be asked the same set of questions about another project.* Yes/No

(PQ35-PQ60) Optional Second Project-Based Response Set

(PQ61-PQ85) Optional Third Project-Based Response Set

Appendix B

Game Testing Survey Questions

B.1 Respondent Information

B.1.1 Information and Consent

Participants were shown the information, contact, and consent sheet for this survey along with a link to download a PDF copy for their records.

(Q1) *Do you consent to participate in this survey?* Yes/No

B.1.2 Eligibility

(Q2) *Do you have at least two years' experience in commercial game development in the past 10 years?* Yes/No

(Q3) *Have you worked for at least one year on a game that has been commercially released?* Yes/No

B.1.3 Demographics

(Q4) *How many years have you been active in the games industry?* Decimal text input between 0–100

(Q5) *Which of the following roles have you previously held in the games industry?* Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner; Other (specify)

(Q6) *How many years of indie game development experience do you have, in the past 10 years?* Integer input between 0–10

(Q7) *How many years of AA game development experience do you have, in the past 10 years?* Integer input between 0–10

(Q8) *How many years of AAA game development experience do you have, in the past 10 years?* Integer input between 0–10

(Q9) *What is your age range?* Under 18; 19–24; 25–34; 35–44; 45–54; 55–65; 65+

(Q10) *What country do you currently reside in?* Text response

B.2 Project-Based Responses

Project Information

(Q11) *Please enter a nickname for the project you have chosen. This is used purely for reference purposes and does not have to be the actual name of the game.* Text response

(Q12) *How would you classify the project?* Indie Game, AA Game, AAA Game

(Q13) *What was your primary role on the project?* Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner; Other (specify)

(Q14) *What were your other roles on the project, if any?* Multi-Select: Developer, Programmer, and/or Engineer; Quality Assurance and/or Game Tester; Team Lead, Producer, and/or Project Manager; Game Designer, Level Designer, Gameplay Designer, and/or Product Owner; Other (specify)

(Q15) *What engine did you use for the project? If it was a proprietary engine that you do not want to identify due to privacy reasons, please enter"Proprietary".* Text response

(Q16) *How long were you working on this project?* 1 year or less; 2–3 years; Over 3 years

(Q17) *What was the size of the team for this project?* 1–49; 50–99; 100–499; 500 or more

(Q18) *What was the approximate budget for this project?* Unknown; Under $1M USD; $1M–$10M USD; Over $10M USD

(Q19) *Did you have a publisher for this project?* Yes, we had a publisher who provided marketing support; Yes, we had a publisher, but they did not provide marketing support; No, we did not have a publisher

(Q20) *How long was this project in development?* 1 year or less; 2–3 years; Over 3 years

Test Performance

(Q21) *Please describe how testing was performed on [this project]. What steps were taken by testing, by whom, and when? Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying.* Open-ended text response

For the following prompts, please use these definitions of different types of tests:

- *Unit Test: A specific test of one aspect or functionality of a class, with a narrow and defined scope. External functionalities and dependencies are mocked out.*
- *Integration Test: A test to ensure the proper interoperability of separate systems.*
- *Smoke Test: Sanity check test that ensures the system does not malfunction under testing and returns normally.*
- *Regression Test: A test to ensure previously-patched bugs do not resurface.*

Respondents rated their agreement with the following prompts on a five-point Likert-like scale: Strongly Disagree, Somewhat Disagree, Neither Agree Nor Disagree, Somewhat Agree, Strongly Agree. They could also select an "N/A" option if the prompt was not applicable to their experience.

(Q22-a) *We primarily used testing tools that we developed in-house*

(Q22-b) *We primarily used third-party testing tools*

(Q22-c) *Team members were able to give actionable input and feedback to the testing process*

(Q22-d) *The majority of testing was done by a dedicated quality assurance team*

(Q22-e) *We routinely ran unit tests on this project*

(Q22-f) *We routinely ran regression tests on this project*

(Q22-g) *We routinely ran integration tests on this project*

(Q22-h) *We routinely ran smoke tests on this project*

Test Planning

(Q23) *Please describe the test planning process for [this project]. How were test plans made, by whom, and when? Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying.* Open-ended text response

Likert-like agreement rating prompts:

(Q24-a) *The testing process was well documented and openly available to the team*

(Q24-b) *The testing process followed the testing plan in most cases without requiring ad-hoc changes*

(Q24-c) *At any given point during development, I had a good idea of the current testing plan including tools, priorities, and schedule*

(Q24-d) *Each major feature for development had a testing or validation plan prior to implementation*

(Q24-e) *There were no concrete testing plans or priorities for the majority of this project*

(Q24-f) *Test plans and priorities were primarily set by developers*

(Q24-g) *Test plans and priorities were primarily set by quality assurance personnel*

(Q24-h) *Test plans and priorities were primary set by supervisors or leads*

(Q24-i) *Test plans or priorities were primarily set by the producer, game director, or senior management*

Testing Goals

(Q25) *Please describe the goals of testing that you and your team had on [this project]. What were you testing for, and how did your testing process accomplish that goal (or fall short)? Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying.* Open-ended text response

Likert-like agreement rating prompts:

(Q26-a) *We routinely performed testing for game mechanics (e.g. game rules, balance between features, overall content design)*

(Q26-b) *We routinely performed testing for technical aspects of the game (e.g. functionality, stability)*

(Q26-c) *We routinely performed testing for user experience (e.g. fun factor, user impression, satisfaction)*

(Q26-d) *The results of testing were often subjective and open to interpretation*

Test Automation

(Q27) *Please describe the degree of manual and automated testing that you experienced on [this project], including whether there were specific areas that were prioritized for testing manually or with automation. Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying.*

Likert-like agreement rating prompts:

(Q28-a) *We used more manual testing than automated testing*

(Q28-b) *We had personnel dedicated to writing scripted or automated tests on this project*

(Q28-c) *The automated testing tools that we used required frequent adjustments as the project progressed*

(Q28-d) *Manual testing yielded more actionable results than automated testing*

(Q28-e) *Automated testing yielded more actionable results than manual testing*

(Q28-f) *Manual testing yielded results that could not be determined through automated testing*

(Q28-g) *Automated testing yielded results that could not be determined through manual testing*

(Q28-h) *Automated testing is important for game development*

(Q28-i) *I am happy with the proportion of automated testing to manual testing that was done for this project*

(Q28-j) *Automated testing was a source of frustration on this project*

Testing Tools

(Q29) *Please list the tools you used for testing on [this project], whether they were used in a certain order, as well as what you perceived as the major benefits and drawbacks to each one (with consideration for functionality and usability).*

Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying. Open-ended text response

Likert-like agreement rating prompts:

(Q30-a) *The tools that we used for testing were developed specifically for games*

(Q30-b) *The tools that we used for testing were intuitive and easy to use*

(Q30-c) *The tools that we used for testing were well-suited to our testing goals*

(Q30-d) *The output from testing was easy to read and understand*

(Q30-e) *The tools that we used for testing complemented each other well*

(Q30-f) *There was a structured approach to testing that was repeated throughout the project*

Test Results

(Q31) *Please describe the format, readability, and usability of the testing output, and how the testing results and output were used by the [project] team.* Open-ended text response

Likert-like agreement rating prompts:

(Q32-a) *The results of testing were used to form actionable plans to address the results*

(Q32-b) *The results of testing could cause major changes to the game's design*

(Q32-c) *The results of testing were used to validate internal targets*

(Q32-d) *The results of testing were primarily used by the testing or development team*

(Q32-e) *The results of testing were used primarily by leads and managers*

(Q32-f) *The results of testing were monitored or visualized over time and used as metrics*

Testing Resources

(Q33) *Please describe your understanding of the allocation of resources for testing on [this project]. What resources were allocated to testing (human resources, funding, time allotment, hardware, etc.) and how?* Open-ended text answer

Likert-like agreement rating prompts:

(Q34-a) *We had sufficient testing resources overall to support this project*

(Q34-b) *We had sufficient testing/QA personnel dedicated to this project*

(Q34-c) *We had sufficient test automation for this project*

(Q34-d) *We had appropriate testing tools given the project and its testing goals*

(Q34-e) *The team requested additional testing resources during the project*

(Q34-f) *Testing was primarily contracted to an outside firm*

(Q34-g) *Testing was primarily performed internally on the team*

(Q34-h) *Testing was done primarily by team members who had other primary tasks, such as developer or designer*

(Q34-i) *Testers were integrated with the development team and had regular direct access to developers*

Additional Information

(Q35) *Would you like to add any additional details about [this project]? Optional open-ended text response*

(Q36) *Would you like to discuss another game project that you worked on for at least one year? If you select "yes", you will be asked the same set of questions about another project.* Yes/No

(Q37-Q62) Optional Second Project-Based Response Set

(Q63-Q87) Optional Third Project-Based Response Set

B.3 General Testing

For the questions on this page, "testing" can be anything to do with the testing process, including requirements, processes, UX, results, documentation, etc. Please be as specific and detailed as possible without compromising privacy and anonymity; you can always obscure details as required if you feel they would be too identifying.

(Q88) *What has made testing easier or faster for you in the past?*

(Q89) *What are the biggest pain points you have encountered with testing?*

(Q90) *What changes would you make or like to see in testing for games?*

Appendix C

Respondents' Demographic Information

Table C.1: Respondents' demographic information

ID	Years in Gamedev	Roles	Years of Indie Exp	Years of AA Exp	Years of AAA Exp	Age	Country
R1	4	Dev; Designer	2	1	1	35–44	Canada
R2	4	QA; Designer	2	1	1	25–34	Canada
R3	16	QA; Lead/Manager	0	0	10	35–44	Canada
R4	10	Dev; QA; Designer	10	0	0	35–44	Canada
R5	10	Dev; Lead; Designer	10	0	0	35–44	Canada
R6	4	QA; Lead/Manager	2	0	2	25–34	United Kingdom
R7	15	QA; Lead/Manager	0	0	10	35–44	Netherlands
R8	6	Dev; Designer	6	0	0	25–34	Canada
R9	3.75	QA; Lead/Manager	1	0	3	25–34	United States
R10	6	Dev; QA; Lead/Manager; Designer	6	0	0	25–34	Canada
R11	9	QA	3	3	3	25–34	Canada
R12	13	QA; Lead/Manager; LiveOps Release/Program Manager	0	1	9	35–44	Canada
R13	3	Designer	2	1	0	25–34	Canada
R14	11	Dev; Lead/Manager; Engineering Manager	3	0	7	25–34	United States
R15	6	Dev; QA; Lead/Manager; Designer; Business/Marketing	6	0	0	25–34	Canada
R16	5	QA; Designer	5	2	0	25–34	United States
R17	1	UX Researcher	0	1	0	25–34	Canada
R18	2	Lead/Manager; Designer	2	0	0	25–34	United States
R19	10	Lead/Manager; Designer	6	0	4	25–34	Canada

Appendix D

Project Information

Table D.1: Project-based information for game submissions

ID	Type	Game Engine	Team Size	Budget	Publisher	Project Duration	Participant Role	Participant on Project
G1-I	Indie	Proprietary	1–49	< $1M	Yes, without marketing	> 3 years	Designer; Dev	2–3 years
G2-N	AAA	Frostbite	100–499	> $10M	Yes, with marketing	3 years	QA	1 year
G3-N	AAA	Unreal	50–99	> $10M	Yes, with marketing	2–3 years	Lead/Manager; QA	2–3 years
G4-I	Indie	Proprietary	1–49	< $1M	Yes, with marketing	> 3 years	Designer; Dev; QA	> 3 years
G5-I	Indie	Unity	1–49	< $1M	No	< 1 year	Lead/Manager; Dev; QA; Designer	1 year
G6-N	AAA	Unreal	100–499	> $10M	Yes, with marketing	> 3 years	QA	2–3 years
G7-N	AAA	Proprietary	100–499	> $10M	Yes, with marketing	> 3 years	Lead/Manager	> 3 years
G8-I	Indie	Unity	1–49	< $1M	No	> 3 years	Dev; Designer	2–3 years
G9-I	Indie	Unity	1–49	< $1M	No	> 3 years	Dev; Lead/Manager	2–3 years
G10-N	AAA	Proprietary	> 500	> $10M	Yes, with marketing	> 3 years	QA	2–3 years
G11-I	Indie	Unity	1–49	< $1M	No	< 1 year	Dev; QA; Lead/Manager; Designer	1 year
G12-N	AAA	Frostbite	> 500	> $10M	Yes, with marketing	> 3 years	QA	1 year
G13-N	AAA	Unreal	100–499	> $10M	Yes, with marketing	> 3 years	Other:LiveOps Program/ Release Manager	> 3 years
G14-N	AA	Unreal	100–499	$1M–$10M	No	2–3 years	Designer	1 year
G15-N	AAA	Unity	100–499	> $10M	No	> 3 years	Dev; Lead/Manager	2–3 years
G16-I	Indie	Unity	1–49	< $1M	No	> 3 years	Dev; QA; Lead/Manager; Designer; Business/Marketing; Writer	> 3 years
G17-I	Indie	Unity	1–49	< $1M	No	2–3 years	Designer; QA	2–3 years
G18-N	AA	Unreal	100–499	Unknown	Yes, without marketing	> 3 years	UX Researcher	1 year
G19-I	Indie	Custom	1–49	< $1M	Yes, with marketing	2–3 years	Lead/Manager; Designer	1 year
G20-I	Indie	Unity	1–49	< $1M	No	> 3 years	Lead/Manager; Designer; Sound Designer	> 3 years
G21-I	AA	Proprietary	1–49	$1–$10M	No	< 1 year	Lead/Manager; Designer	1 year
G22-N	AAA	Unreal	100–499	> $10M	Yes, without marketing	> 3 years	Designer; Dev	> 3 years

Appendix E

Distributions of Likert-like responses per prompt

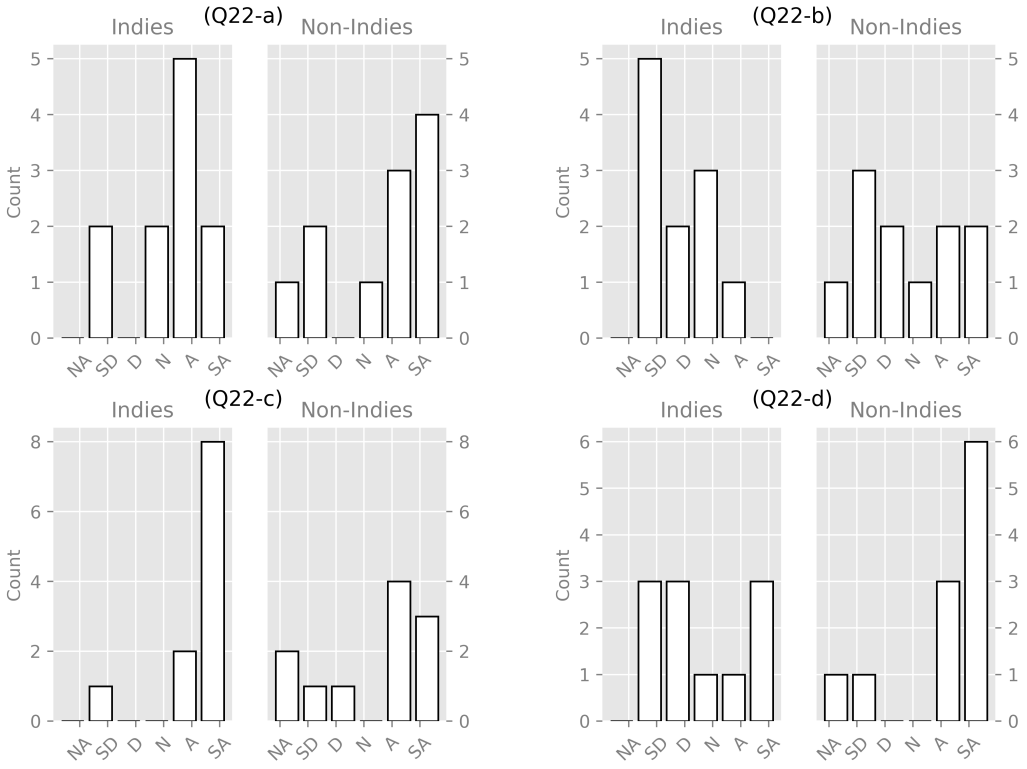
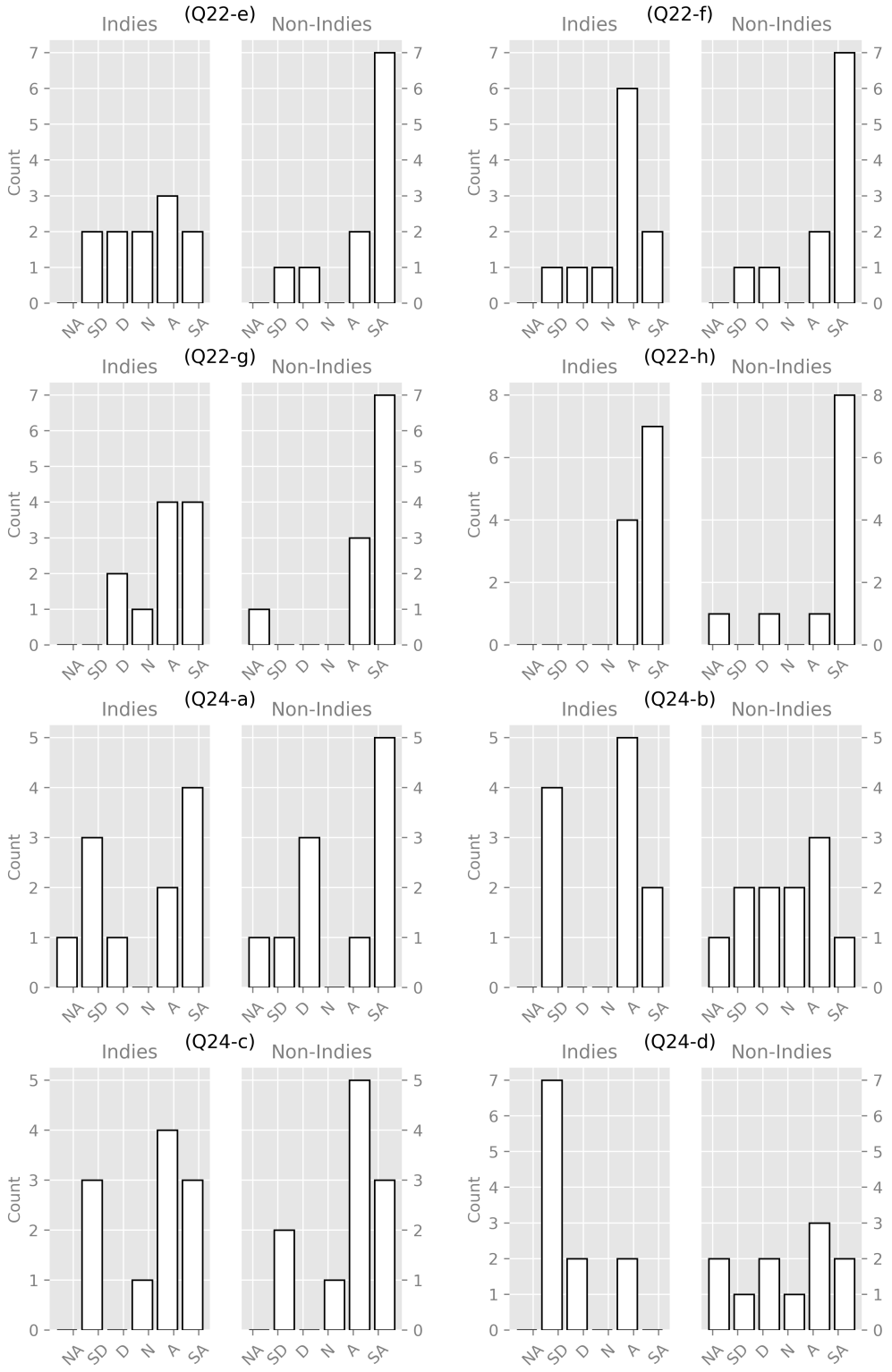
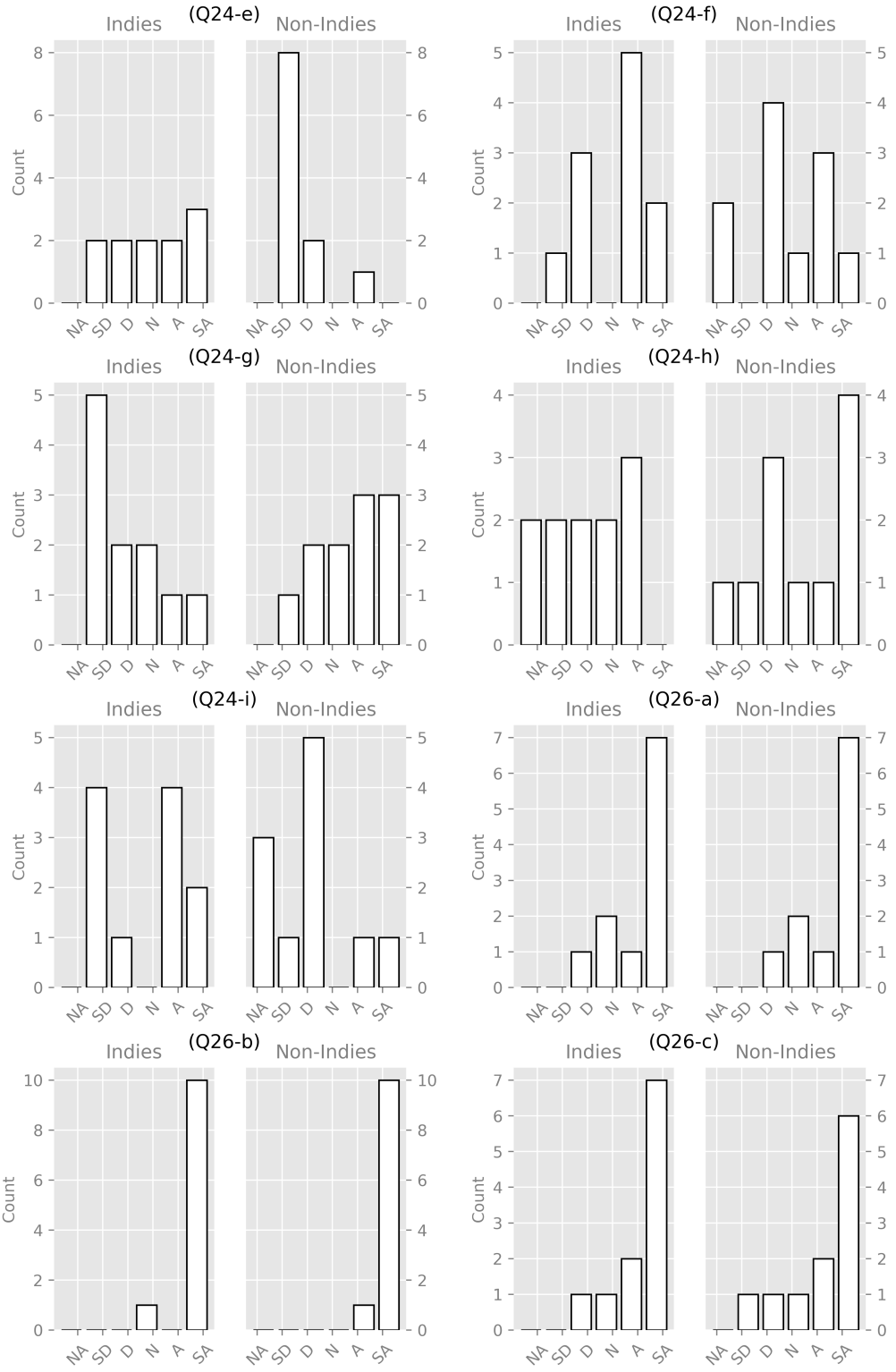
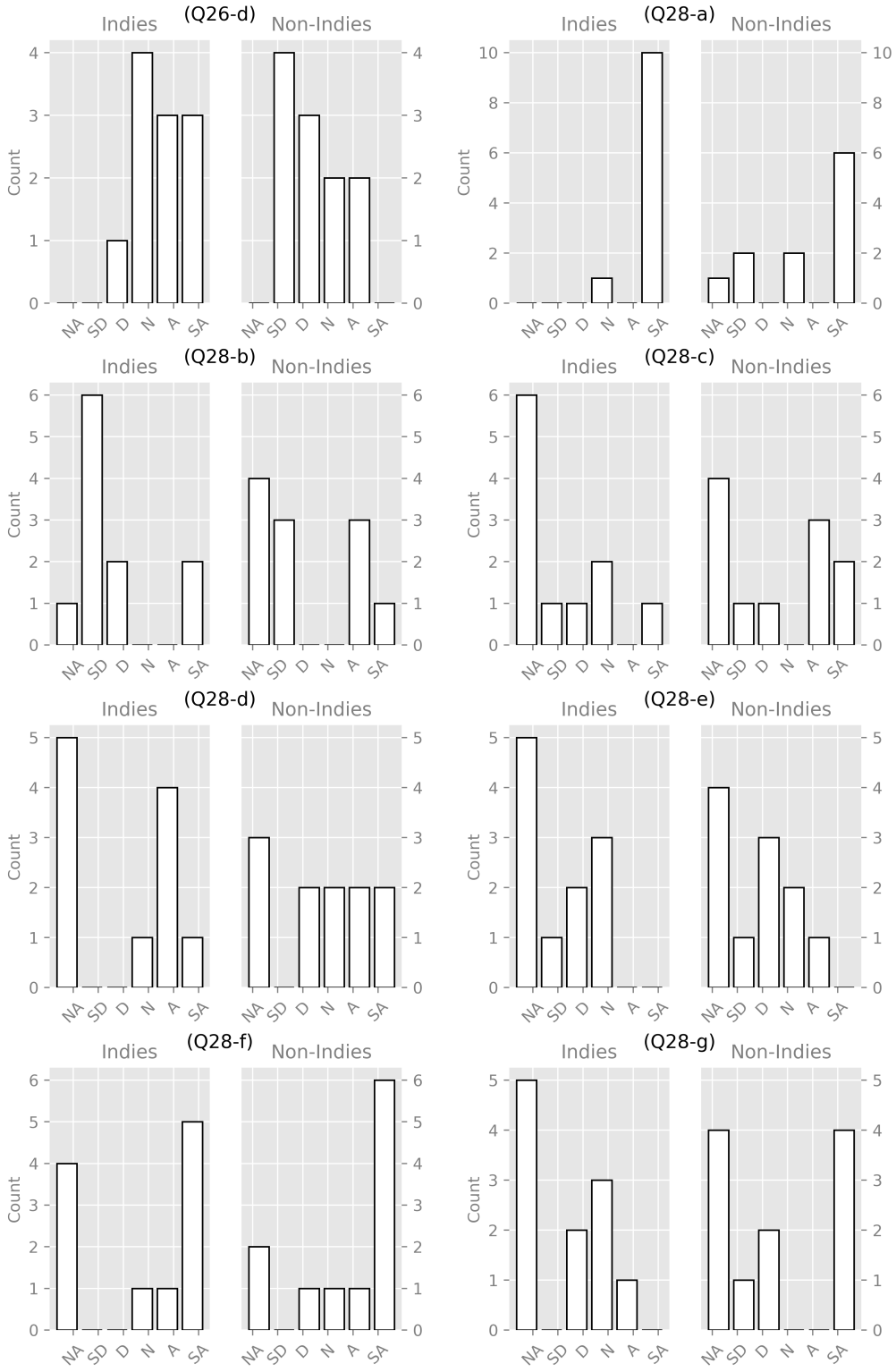
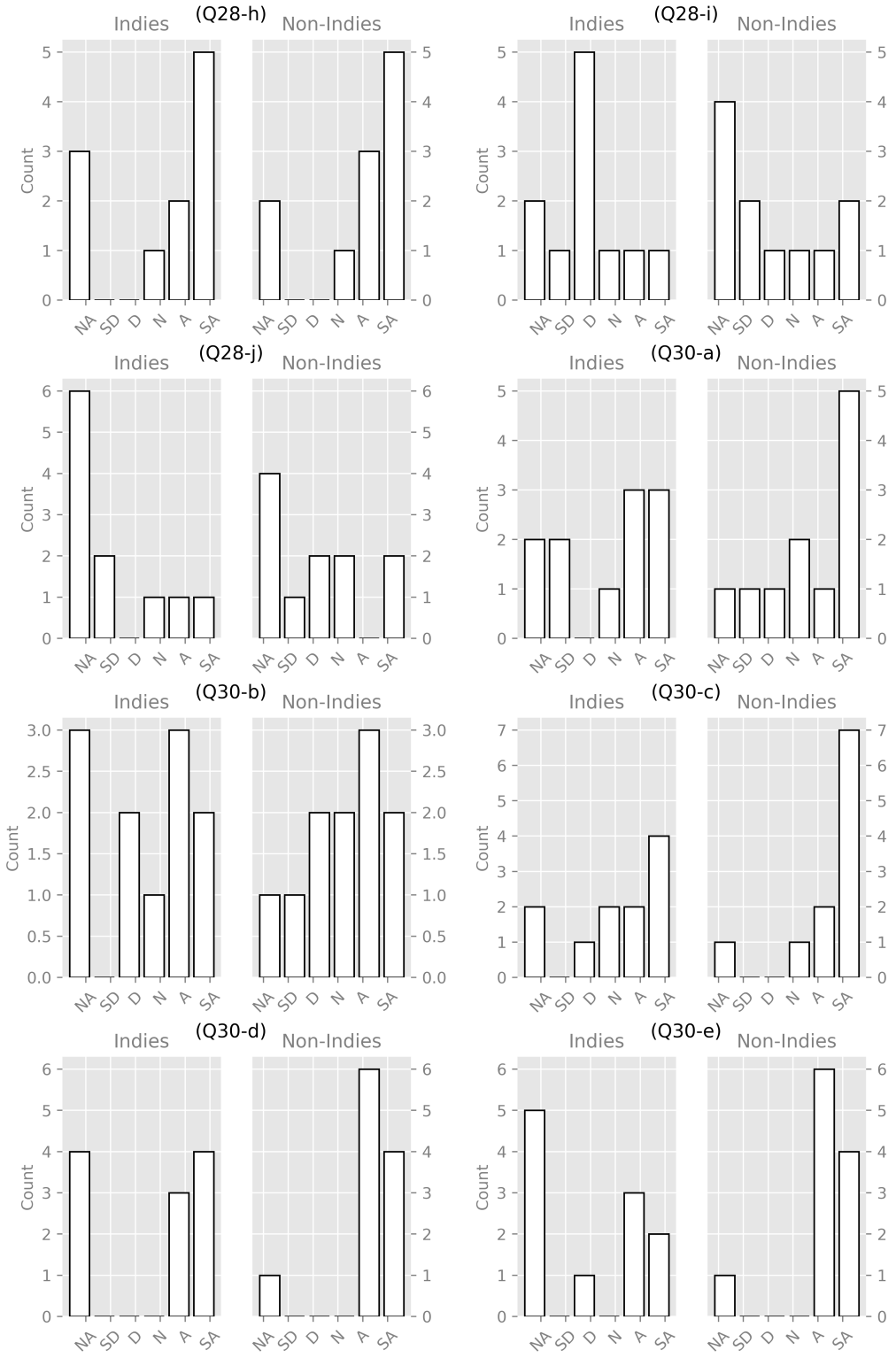


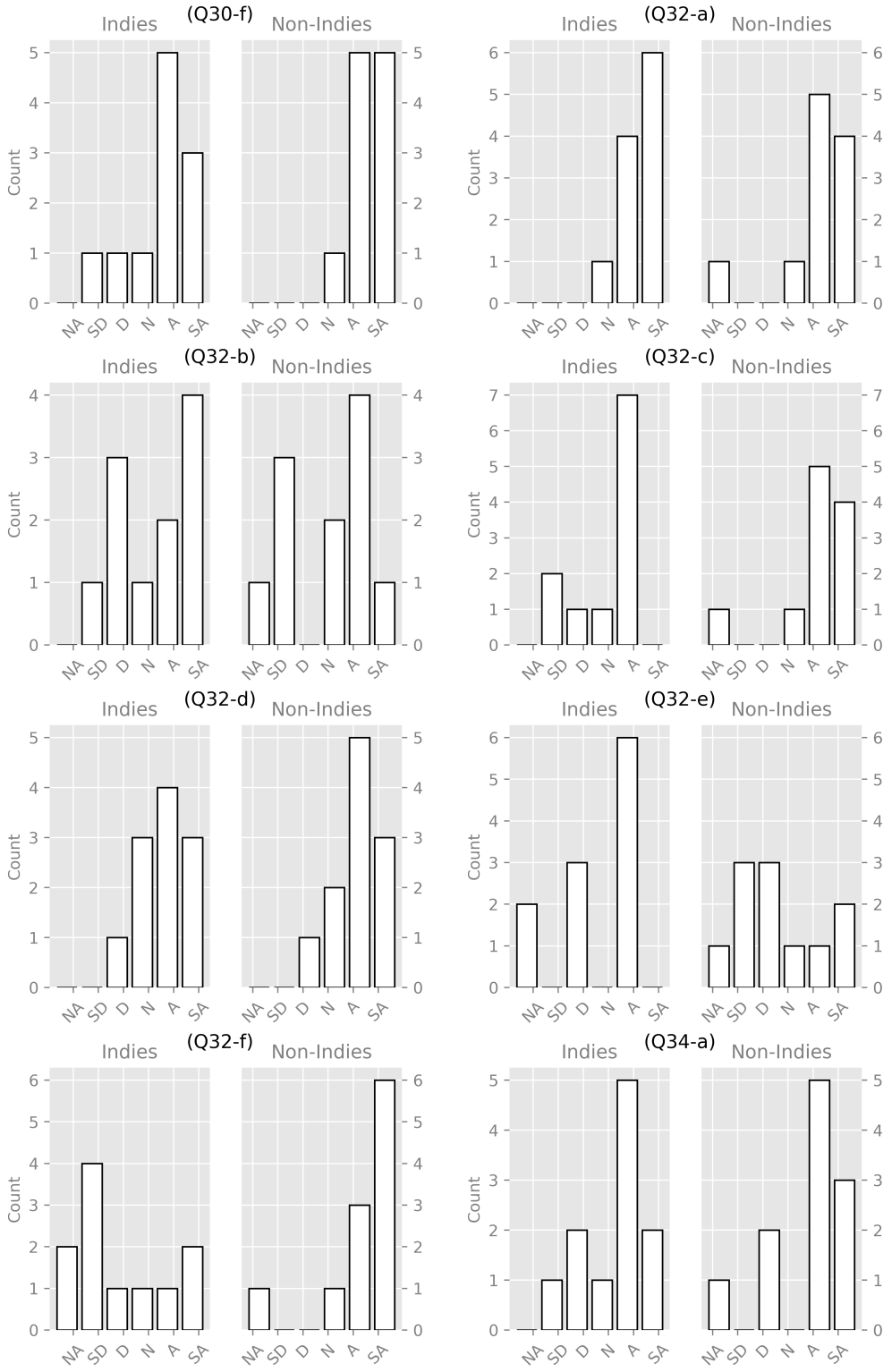
Figure E.1: Distributions of Likert-like responses per prompt

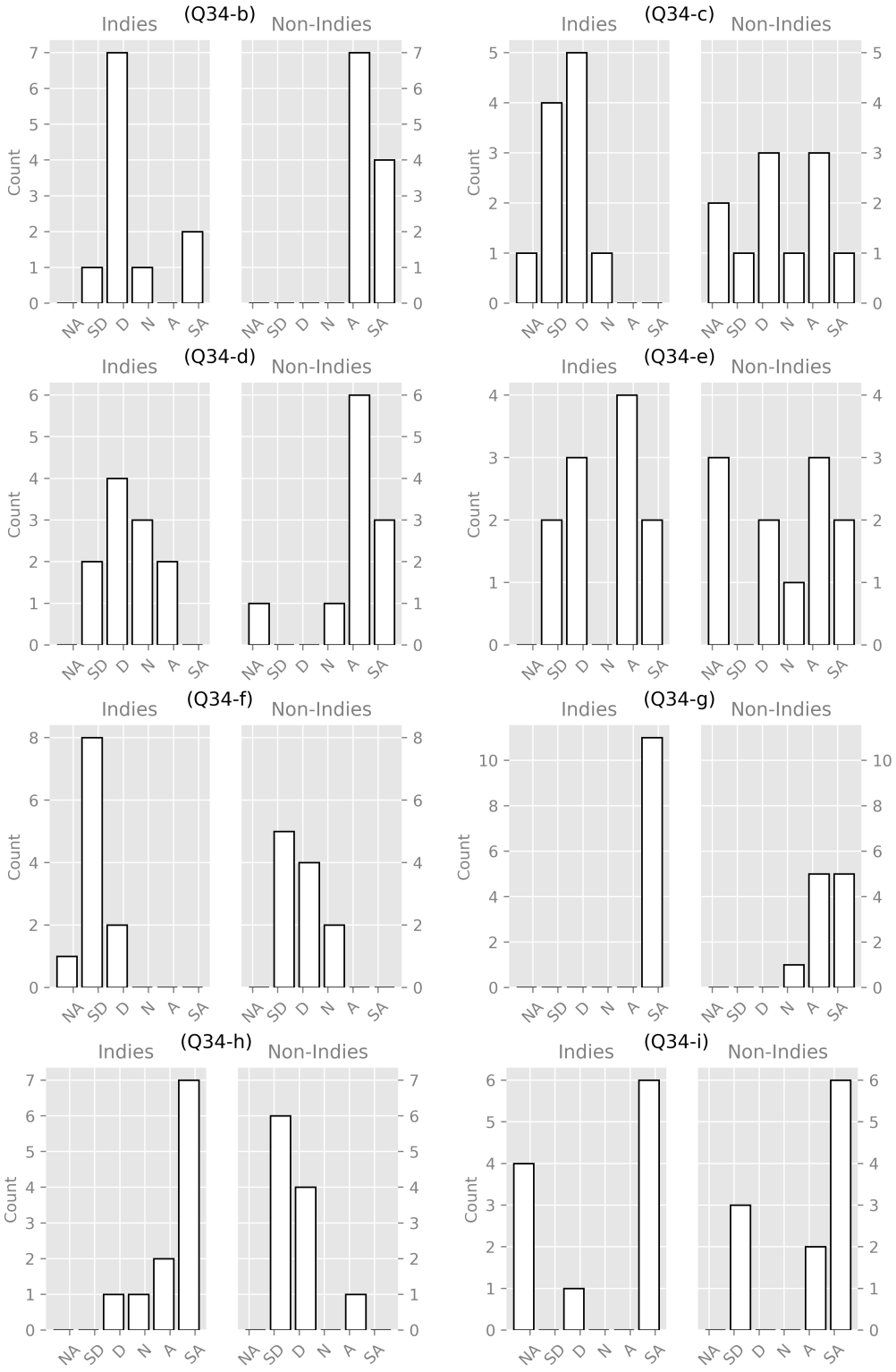

(Q22-e)
Indies
Non-Indies
(Q22-f)
Indies
Non-Indies
(Q22-g)
Indies
Non-Indies
(Q22-h)
Indies
Non-Indies
(Q24-a)
Indies
Non-Indies
(Q24-b)
Indies
Non-Indies
(Q24-c)
Indies
Non-Indies
(Q24-d)
Indies
Non-Indies
Count
NA
SD
D
N
A
SA


(Q24-e)
Indies
Non-Indies
Count
NA SD D N A SA
(Q24-f)
Indies
Non-Indies
Count
NA SD D N A SA
(Q24-g)
Indies
Non-Indies
Count
NA SD D N A SA
(Q24-h)
Indies
Non-Indies
Count
NA SD D N A SA
(Q24-i)
Indies
Non-Indies
Count
NA SD D N A SA
(Q26-a)
Indies
Non-Indies
Count
NA SD D N A SA
(Q26-b)
Indies
Non-Indies
Count
NA SD D N A SA
(Q26-c)
Indies
Non-Indies
Count
NA SD D N A SA


(Q26-d)
Indies
Non-Indies
Count
NA SD D N A SA
(Q28-a)
(Q28-b)
(Q28-c)
(Q28-d)
(Q28-e)
(Q28-f)
(Q28-g)


(Q28-h)
Indies
Non-Indies
Count
NA
SD
D
N
A
SA
(Q28-i)
(Q28-j)
(Q30-a)
(Q30-b)
(Q30-c)
(Q30-d)
(Q30-e)


(Q30-f)
Indies
Non-Indies
(Q32-a)
(Q32-b)
(Q32-c)
(Q32-d)
(Q32-e)
(Q32-f)
(Q34-a)
Count
NA
SD
D
N
A
SA


(Q34-b)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-c)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-d)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-e)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-f)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-g)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-h)
Indies
Non-Indies
Count
NA SD D N A SA
(Q34-i)
Indies
Non-Indies
Count
NA SD D N A SA

Appendix F

Boxplots of Likert-like responses per prompt

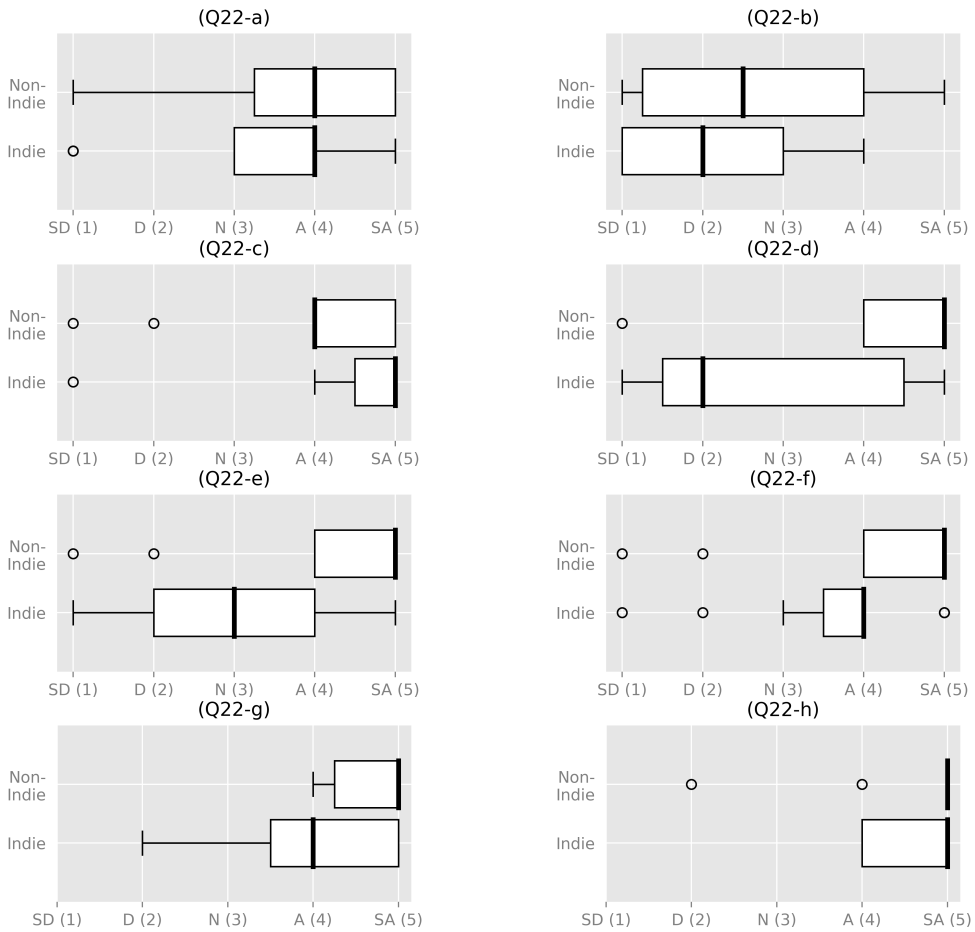
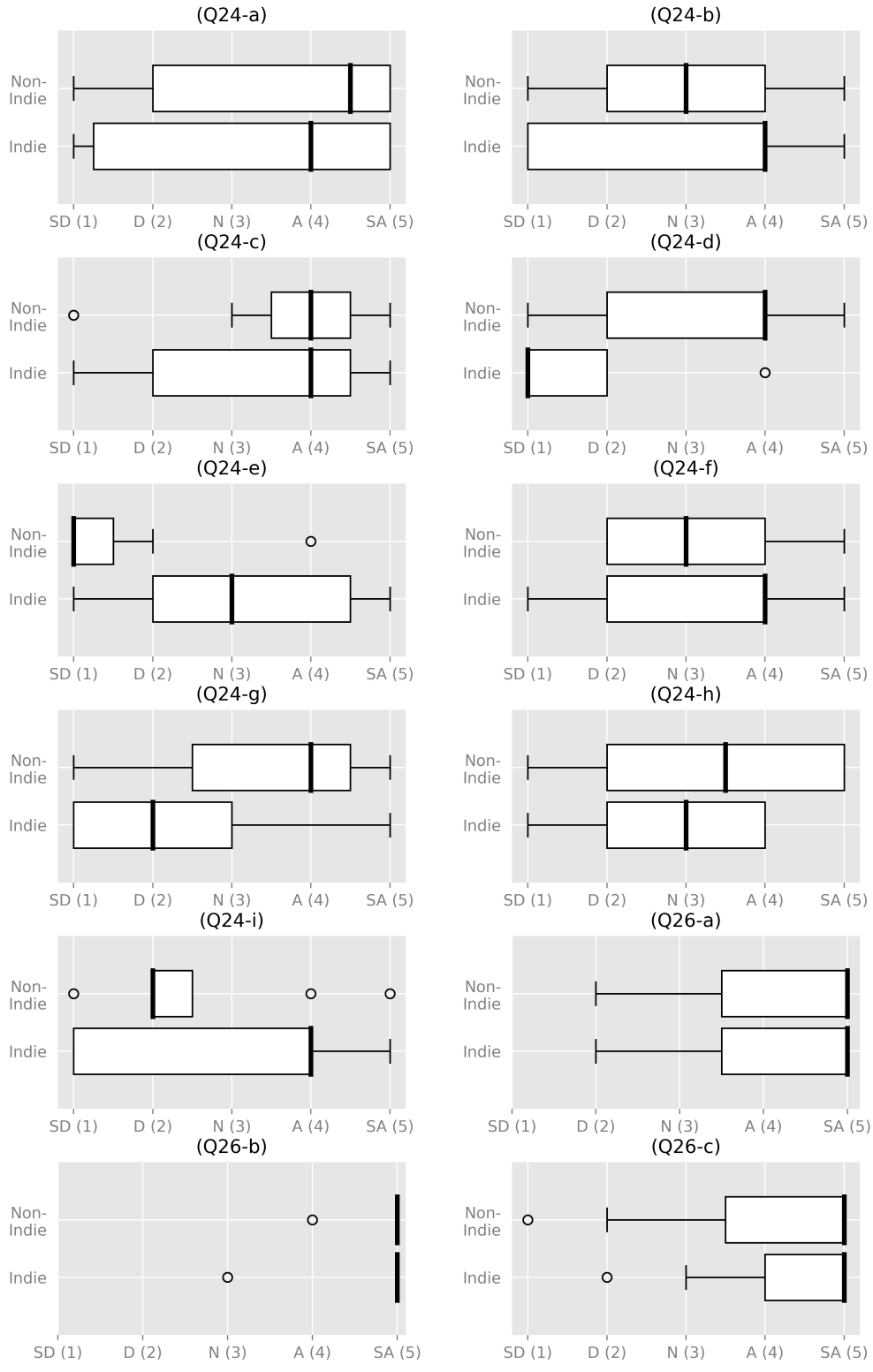
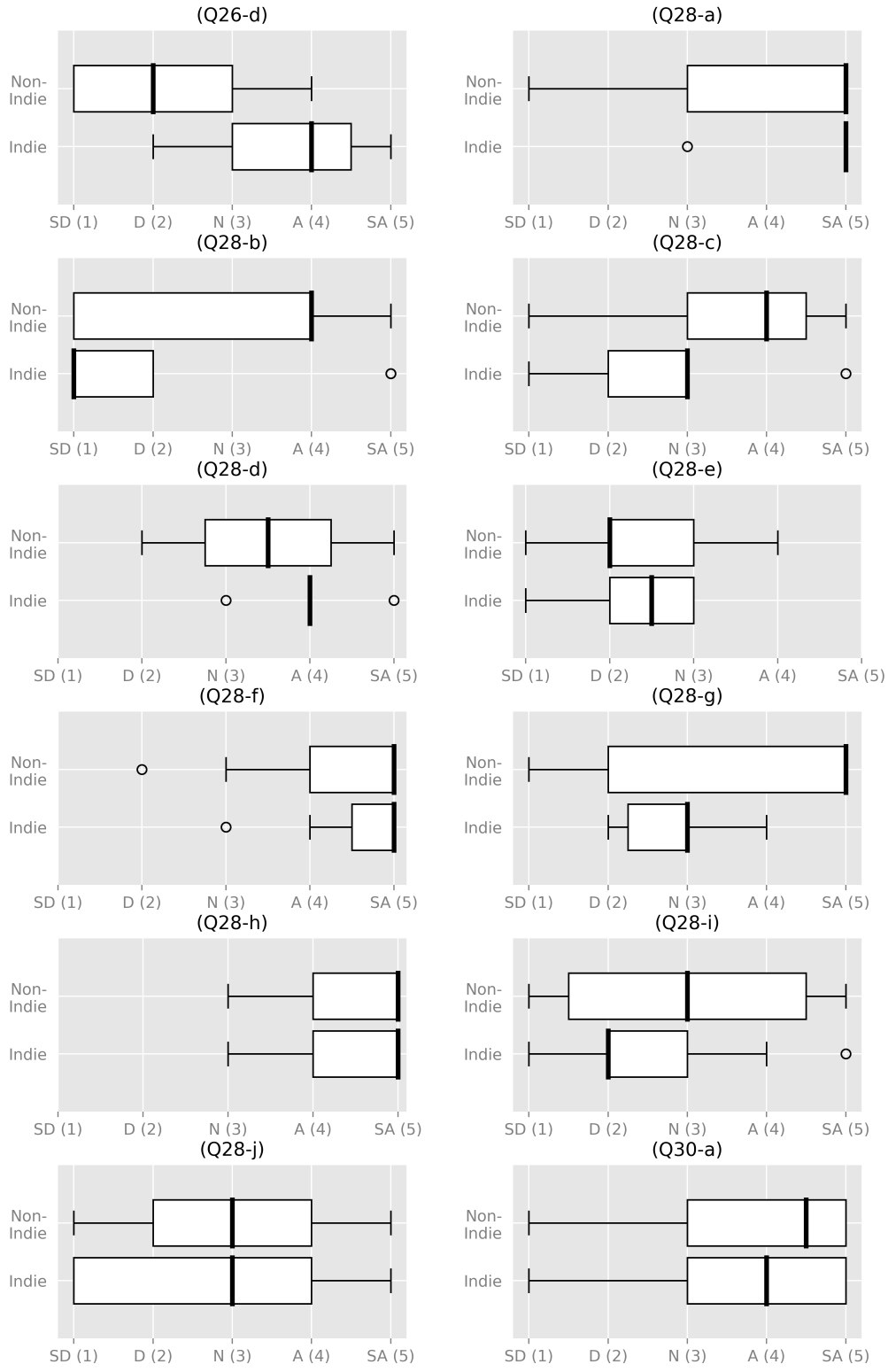
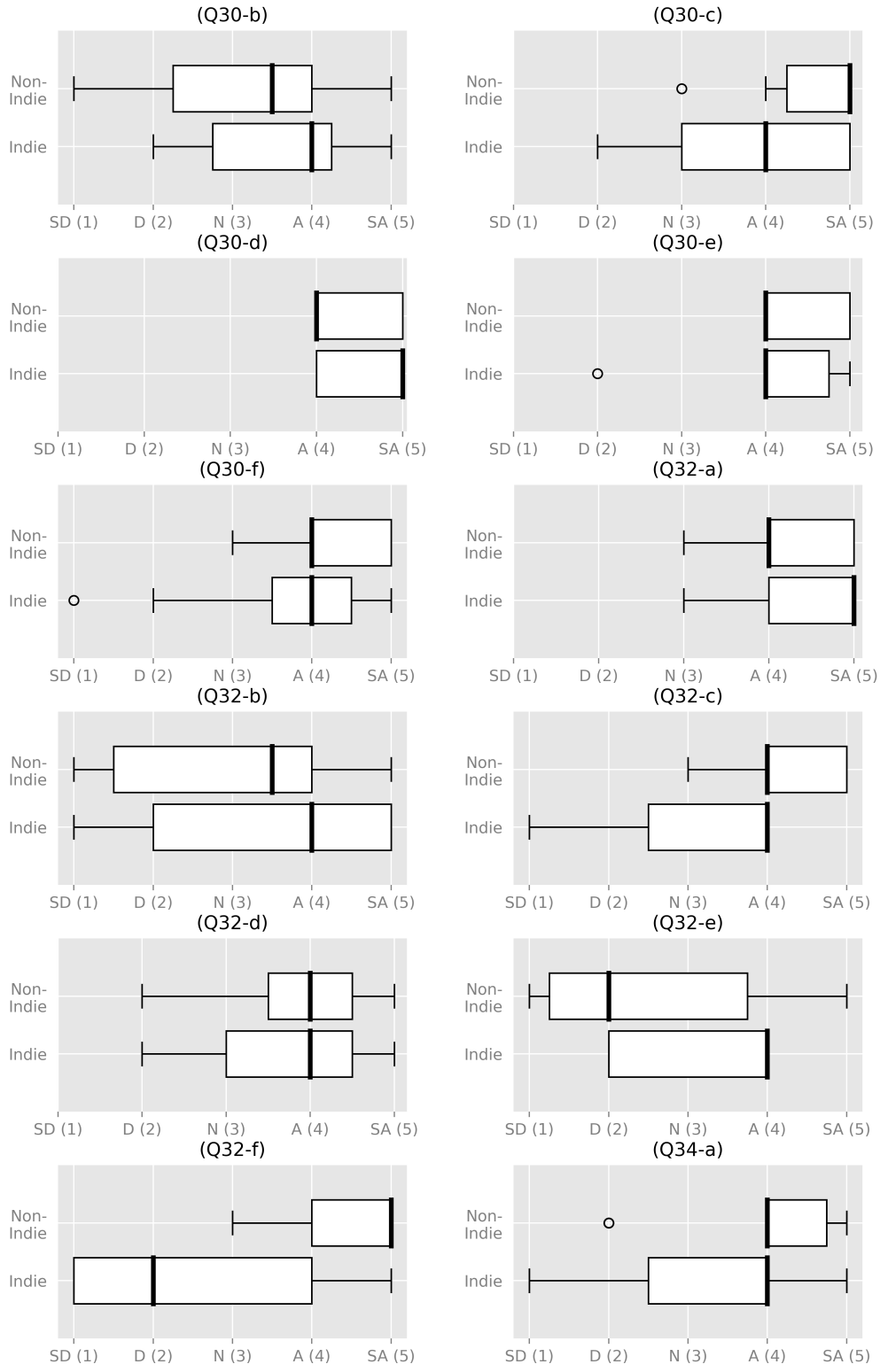
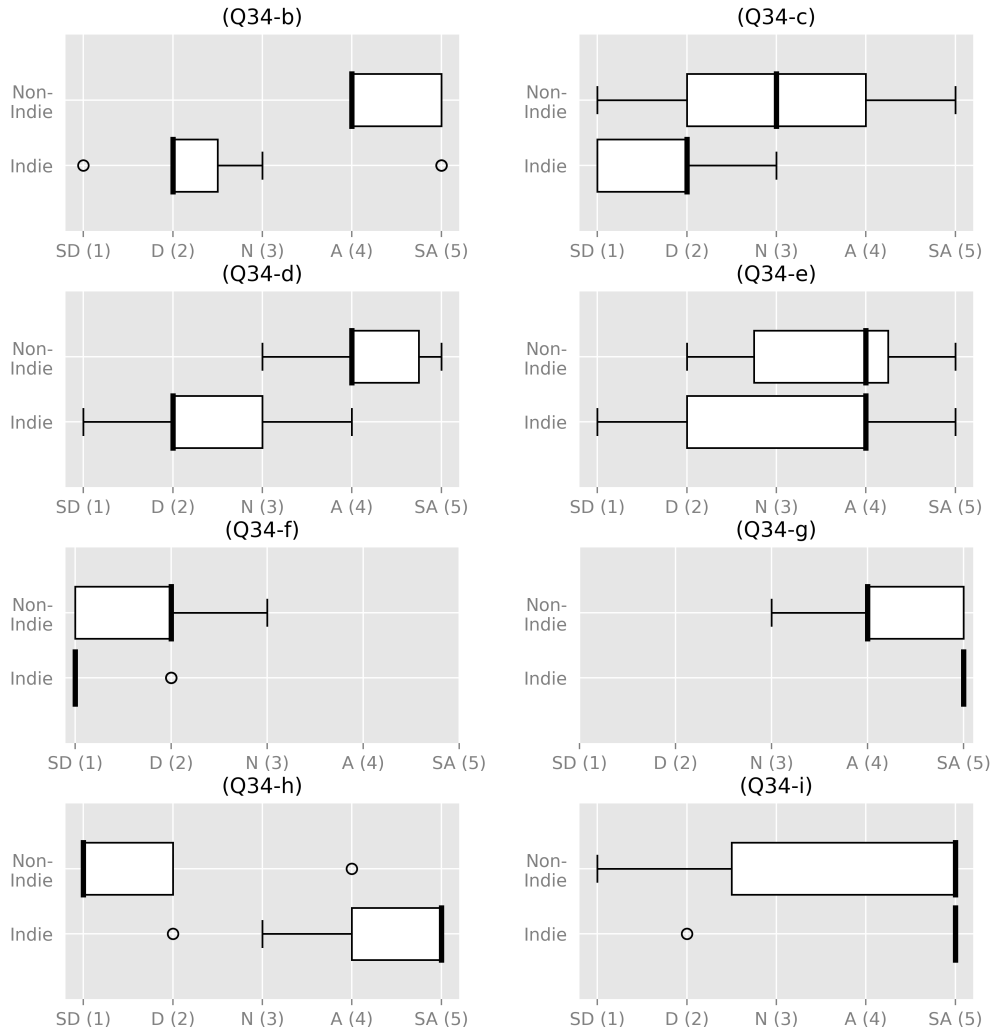


Figure F.1: Boxplots of Likert-like responses per prompt


(Q24-a)
(Q24-b)
(Q24-c)
(Q24-d)
(Q24-e)
(Q24-f)
(Q24-g)
(Q24-h)
(Q24-i)
(Q26-a)
(Q26-b)
(Q26-c)
Non-Indie
Indie
SD (1)
D (2)
N (3)
A (4)
SA (5)


(Q26-d)
(Q28-a)
(Q28-b)
(Q28-c)
(Q28-d)
(Q28-e)
(Q28-f)
(Q28-g)
(Q28-h)
(Q28-i)
(Q28-j)
(Q30-a)
Non-Indie
Indie
SD (1)
D (2)
N (3)
A (4)
SA (5)


(Q30-b)
(Q30-c)
(Q30-d)
(Q30-e)
(Q30-f)
(Q32-a)
(Q32-b)
(Q32-c)
(Q32-d)
(Q32-e)
(Q32-f)
(Q34-a)
Non-Indie
Indie
SD (1)
D (2)
N (3)
A (4)
SA (5)


(Q34-b)
(Q34-c)
(Q34-d)
(Q34-e)
(Q34-f)
(Q34-g)
(Q34-h)
(Q34-i)
Non-Indie
Indie
SD (1)
D (2)
N (3)
A (4)
SA (5)