Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2025-IJ-005

2025-IJ-005

A Mixed-Methods Studyof Model-Based GUI Testing in Real-World Industrial Setings

Shaoheng Cao, Renyi Chen,Wenhua Yang, Minxue Pan, Xuandong Li

Technical Report 2025

A Mixed-Methods Study of Model-Based GUI Testing in Real-World Industrial Settings

SHAOHENG CAO, Nanjing University, China
RENYI CHEN, Samsung Electronics (China) R&D Centre, China
WENHUA YANG*, Nanjing University of Aeronautics and Astronautics, China
MINXUE PAN*, Nanjing University, China
XUANDONG LI, Nanjing University, China

Model-based testing (MBT) has been an important methodology in software engineering, attracting extensive research attention for over four decades. However, despite its academic acclaim, studies examining the impact of MBT in industrial environments—particularly regarding its extended effects—remain limited and yield unclear results. This gap may contribute to the challenges in establishing a study environment for implementing and applying MBT in production settings to evaluate its impact over time. To bridge this gap, we collaborated with an industrial partner to undertake a comprehensive, longitudinal empirical study employing mixed methods. Over two months, we implemented our MBT tool within the corporation, assessing the immediate and extended effectiveness and efficiency of MBT compared to script-writing-based testing. Through a mix of quantitative and qualitative methods—spanning controlled experiments, questionnaire surveys, and interviews—our study uncovers several insightful findings. These include differences in effectiveness and maintainability between immediate and extended MBT application, the evolving perceptions and expectations of engineers regarding MBT, and more. Leveraging these insights, we propose actionable implications for both the academic and industrial communities, aimed at bolstering confidence in MBT adoption and investment for software testing purposes.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; • **General and reference** → **Empirical studies**.

Additional Key Words and Phrases: Model-based testing, empirical studies, GUI testing.

ACM Reference Format:
Shaoheng Cao, Renyi Chen, Wenhua Yang, Minxue Pan, and Xuandong Li. 2025. A Mixed-Methods Study of Model-Based GUI Testing in Real-World Industrial Settings. *Proc. ACM Softw. Eng.* 2, FSE, Article FSE070 (July 2025), 22 pages. https://doi.org/10.1145/3715789

1 Introduction

Model-Based Testing (MBT) has attracted considerable research interest for over four decades. This methodology leverages system models to facilitate various testing activities, such as the generation of test cases, creation of test data, and formulation of test oracles [16]. Since its emergence, MBT

*Corresponding authors.

Authors' Contact Information: Shaoheng Cao, shaohengcao@smail.nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China; Renyi Chen, renyi88.chen@samsung.com, Samsung Electronics (China) R&D Centre, Nanjing, China; Wenhua Yang, ywh@nuaa.edu.cn, College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China; Minxue Pan, mxp@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China; Xuandong Li, lxd@nju.edu.cn, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.

ACM 2994-970X/2025/7-ARTFSE070
https://doi.org/10.1145/3715789

has been vigorously advocated for and explored within the academic community. A myriad of studies [4, 6, 9, 15, 28, 30, 31] has been undertaken to evaluate the pragmatic effectiveness of MBT. For example, empirical investigations have assessed MBT's applicability to certain software systems, such as Siemens healthcare software [28]. These studies have underscored MBT's effectiveness in specific testing contexts, yet they fall short of offering a comparative assessment against testing methodologies that are still broadly utilized in software system testing, such as manual script writing.

To address this gap, several empirical investigations have sought to explore the distinctions between MBT and conventional testing approaches within certain software systems. However, these studies are seldom carried out fully in industrial production environments. For instance, some research lacks any industrial context [7], while other studies implement MBT or comparative testing techniques in controlled laboratory environments [12, 22, 29, 34]. The lack of data from actual industrial production environments leads to a conservative view within the industry regarding the effectiveness and efficiency of MBT in real-world production settings [16]. An inconvenient truth is that, although MBT has been proposed for a long time, it can conservatively be assumed that MBT has not been widely adopted within the industry. Consequently, there is a pressing necessity to investigate MBT's effectiveness within an entirely real industrial environment and to benchmark it against other testing methodologies currently in use in industrial settings.

This is a challenging task since testing is an ongoing and evolving component of the software development lifecycle. Hence, with MBT serving as a testing strategy, the investigation should consider not only its immediate effectiveness and efficiency but also its extended influence over time. A critical aspect frequently underscored in MBT research is the substantial time investment required for manual model construction. Some researchers argue that this preparation process bolsters the comprehension of the System Under Test (SUT) [17], thereby potentially heightening fault detection effectiveness and efficiency [31]. Moreover, the initial model-building efforts are considered foundational, necessitating only sporadic updates for future SUT versions [8, 28]. Yet, the concrete benefits of these initial endeavors are somewhat ambiguous in the existing research literature. The degree to which such efforts, particularly in model creation, contribute to extended testing benefits (e.g., exhaustive system coverage) remains an open question within both academic and industrial circles. This uncertainty contributes to the restrained deployment of MBT in the industry [2, 11] and underscores a notable gap in existing research. Many previous studies feature a restricted, immediate focus and lack comprehensive, extended evaluations of MBT's effect on system testing [22, 29, 34]. Therefore, there is a need to conduct a longitudinal empirical study to evaluate MBT's effectiveness and efficiency against other testing methods and to assess whether its benefits justify the initial investment.

Furthermore, the promotion and application of MBT in industrial production environments are influenced not only by MBT's own technical factors but also by non-technical factors, including those related to human, organization, management, and cost aspects. These elements collectively shape the decision to adopt or discard the MBT technique in practice. Within specific industrial contexts, these non-technical factors are indispensable to a comprehensive understanding and should not be neglected, as is often the case in existing research. Therefore, it is necessary to understand industry practitioners' perspectives on MBT, exploring their experiences and insights.

To summarize, the current MBT research is notably lacking in the following valuable elements: (1) the use of a fully industrial setting for both MBT and comparison methods, (2) an extended evaluation of MBT's effectiveness and efficiency over a certain time period, and (3) the exploration of both technical and non-technical factors influencing MBT adoption.

To fill the gaps, this study conducts a comprehensive comparison between MBT and the script-writing testing approach within a full industrial context. It utilizes a longitudinal design to examine

both the immediate and extended effects of MBT. In addition to quantitative comparison, a mixed-methods approach, encompassing both questionnaire surveys and interviews, is employed to investigate the technical and non-technical factors affecting MBT's adoption in real-world industrial production settings. The study was carried out with our industrial collaborator, the international corporation Samsung, involving the implementation of MBT on its GUI applications for the Tizen platform. Tizen is an open-source, Linux-based operating system with the capability to connect to a diverse range of devices, including smart TVs, smartphones, wearables, and IoT devices. Our specific focus on conducting MBT experiments on the Tizen applications was driven by the widespread prevalence of these applications, which run on over 30 million smart TVs globally. We anticipate that conducting MBT experiments on these Tizen applications will provide insights that effectively mirror real-world industrial scenarios.

Specifically, our MBT tool is based on Interaction Flow Modeling Language (IFML) [3], a standardized modeling language endorsed by the Object Management Group. IFML serves as a means to describe the graphical user interface (GUI), user interaction, and control behavior of software applications. With moderate extensions, our IFML-based framework can be applied for model-based testing of Tizen applications (more details in Section 2). We conducted immediate and extended evaluations to assess the effectiveness and efficiency of applying MBT to Tizen applications, benchmarking it against conventional manual scripting methods. Moreover, a questionnaire survey was administered to engineers at our industrial collaborator to elicit their views on manual scripting versus MBT, complemented by interviews with these engineers to gain an in-depth understanding of their perceptions concerning the challenges of deploying MBT in an industrial context, thereby shedding light on the non-technical aspects crucial for MBT's successful adoption.

Our study yields several insightful findings. First, MBT, upon initial introduction, does not exhibit superiority over conventional test automation techniques, challenging the assertions of some prior research. Its effectiveness, rather, proves to be comparable. Second, the extended application of MBT demonstrates distinct advantages in efficiency, test maintainability, and the reduction of manual labor. This outcome highlights the predominant benefits of MBT in the long run, advocating for its extended use. Third, although initial user perceptions do not distinguish between the effectiveness of MBT and conventional methods, these views significantly evolve, recognizing MBT's enhanced effectiveness and efficiency over time. Lastly, contrary to expectations, study participants primarily identified MBT's chief advantage as not an increase in test coverage but a notable boost in work efficiency and a decrease in workload. This perspective emphasizes MBT's operational benefits, which surpass mere technical metrics like coverage. From these insights, we derive valuable lessons and research opportunities for implementing MBT in industry.

Overall, the main contributions of this paper are as follows:

- We conducted a comprehensive study within a full industrial environment to compare the effectiveness and efficiency of model-based testing and script-writing-based testing, examining both the immediate and extended effects.
- We enriched our study with questionnaire surveys and interviews to understand both the technical and non-technical factors that influence MBT's implementation in industrial production settings.
- We compiled diverse sources of data to derive distinctive insights, providing in-depth analyses and implications tailored to the interests and requirements of both the academic community and industry professionals.

The remainder of the paper is structured as follows. Section 2 provides an overview of the study context and the details of our MBT approach. Section 3 outlines the methodology employed in our study. The results and findings of this study are presented in Section 4. Lessons learned are

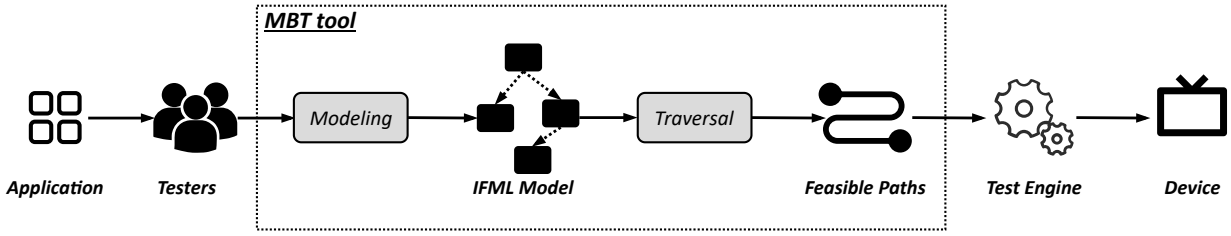

Fig. 1. Testing Pipeline with the MBT Tool.

discussed in Section 5, while Section 6 addresses the threats to the validity of the study. Section 7 explores related work, and finally, Section 8 summarizes the paper.

2 Study Context

This section offers an overview of the industrial background and the motivations behind this study. It then delves into the specifics of the MBT methodology employed to test Tizen applications on smart TVs.

2.1 Industrial Background and Motivations

In the domain of application development, the conventional testing approach required testers to manually write scripts for automated testing. This process required the manual composition of test cases to cover various scenarios, which were then converted into executable scripts by testing frameworks. Despite automation in executing test actions through scripts, the process remained significantly labor-intensive and time-consuming. Such challenges prompted the search for more innovative testing automation methods to reduce costs and enhance test efficiency. MBT emerged as a viable alternative, and the corporation began to experiment with the MBT approach for testing. The adoption of MBT marked a significant shift in this landscape. With MBT, the testing process now revolves around the creation of models for each application, enabling the automation of both test case generation and execution. Importantly, constructing these models represents a one-time investment. Testers only need to make updates to the model as applications are updated.

2.2 The MBT Approach

Figure 1 illustrates the testing pipeline implemented with our MBT tool. This tool is built upon the Eclipse Modeling Framework, offering a graphical user interface for testers to model the GUI, user interactions, and control behaviors of the SUT. After the models are constructed, the tool navigates through them to identify feasible paths that represent potential sequences of user interactions. These sequences, regarded as test cases, are then translated into test scripts according to the test engine's APIs and executed on real devices. We developed this custom MBT tool because existing open-source tools do not fully support the modeling of smart TV applications, and domain-specific semantics must be incorporated into modeling languages to generate executable test cases. For example, smart TV user interactions are controller-based, with only the widget under the controller's focus enabled, a behavior not adequately handled by current tools. The development of a custom MBT tool ensures that the entire MBT workflow aligns with MBT methodology. Next, we will elaborate on the details of the MBT tool.

2.2.1 Extended IFML. Our MBT approach for testing Tizen applications is developed based on the IFML [3]. The original IFML is designed to support the modeling of GUIs and their interaction logic for applications. It defines a set of core concepts that are common to GUI-driven applications,

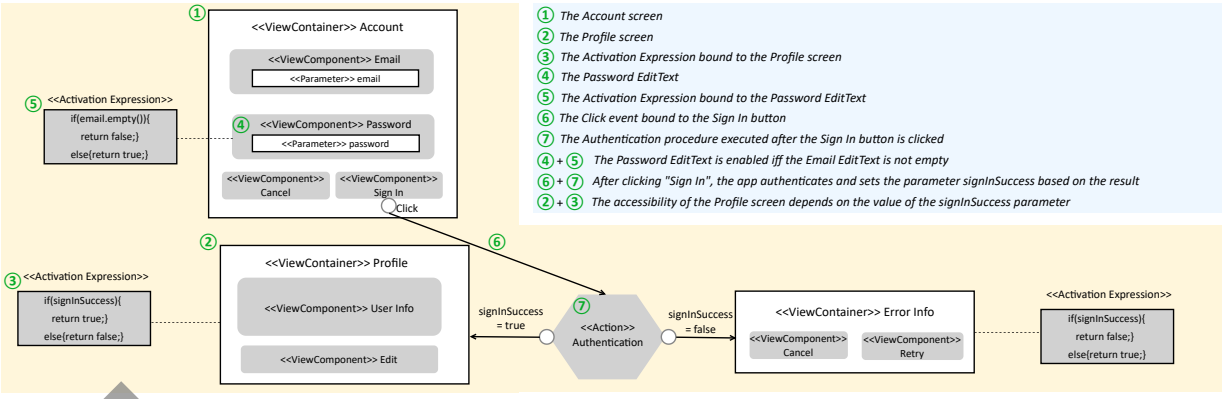

Fig. 2. A T-IFML Example of the Sign In Scenario.

alongside extension mechanisms designed to refine these concepts' semantics further. There are several foundational concepts within IFML.

- *ViewContainer*: Serves as the structure that accommodates various GUI elements, such as grid layouts. Elements within a container often share certain similarities across functionality or appearance.
- *ViewComponent*: Represents the fundamental units of user interaction, including buttons, icons, images, or any other interactive GUI elements in the application.
- *Event*: Refers to the operations that can be executed on a *ViewComponent* or *ViewContainer*. *ClickEvent*, for instance, is the most common one.
- *Expressions* and *Actions*: These concepts are designed to express the internal logic of the applications to be modeled. While *Actions* define specific behaviors, *Expressions* describe the logic behind those behaviors.

Due to the variability and unique semantics of different applications, we extended the original IFML semantics. To be specific, we moderately expanded upon the foundational concepts of IFML to create a Tizen platform-specific extension named T-IFML. This expansion includes additional types of containers, components, and events. Furthermore, we have enriched the *Expressions* and *Actions*, enabling the users to write C-like code snippets for expressions. This enhancement seeks to better align the model with the practical constructs of the application. The design of the T-IFML bridges the gap between the model and the actual application, thus reducing the cognitive effort required to understand the relationship between the two. Figure 2 shows a simplified T-IFML example of the sign-in scenario. Users begin at the Account screen where they input the email and password and then proceed to click the sign-in button. Following this action, the system conducts an authentication procedure. If the authentication is successful, the application navigates to the Profile screen. In contrast, if authentication fails, the user is redirected to the Error screen.

2.2.2 Test Case Generation. To generate test cases for applications from the T-IFML model, we designed a path traversal algorithm. These applications are characterized by their controller-based interaction nature. At any given time, a single component on the screen is designated as "focused". To interact with a different component, users must navigate the focus to the target component using the remote controller. The path traversal algorithm aims to generate all the feasible paths that reach all the different components across diverse application states.

Here, we briefly outline the idea of the path traversal algorithm. The algorithm takes an application's T-IFML model as input, which is structured as a graph comprising nodes and edges, denoted as V and E, respectively. The nodes can either be *ViewComponents* or *Actions*, as defined by the

IFML semantics, while the edges describe transitions between nodes, representing the interactions triggered by specific events. Each node v and edge e is associated with specific constraints, denoted as $v.C$ and $e.C$, described by the model's *Expressions*. These constraints are crucial for defining the permissible interactions and states within the application.

The algorithm begins at the default focused node $v_0 \in V$ and systematically traverses the model, generating all possible paths that connect v_0 to every other node in the model. These paths, collectively referred to as P, represent potential test cases that can be used to test different functionalities of the application. However, not each path $p \in P$ is feasible. For instance, as illustrated in Figure 2, the Password *ViewComponent* is interactive only after the email field is filled; otherwise, it remains disabled. This condition is governed by an activation expression. Therefore, a path leading to the Password component without first inputting an email is infeasible and should not be considered a valid test case.

To filter out such infeasible paths and optimize the test cases, the algorithm proceeds to check the satisfiability of each path $p \in P$. This is done in two stages: first, for each path p, its associated nodes and edges, denoted as $p.V$ and $p.E$, are used to gather the constraints, $p.V.C$ and $p.E.C$, collectively referred to as $p.C$. In the second stage, the algorithm checks the satisfiability of $p.C$. If a path fails this check, it is infeasible and should be removed from P. This process ascertains the feasibility of the generated paths. The feasible paths are then translated into operational sequences, which are used to test the specific functionalities of the application.

3 Study Methodology

In this section, we provide a detailed exposition of our research questions, the design and execution of our study, as well as the procedure for data collection and analysis.

3.1 Research Questions

To investigate the application of the MBT methodology within an industrial environment, aiming to assess its effectiveness as well as efficiency and identify factors influencing MBT's practical use across real-world software projects and stakeholders, we propose the following research questions:

- **RQ1**: How does MBT perform against conventional testing automation methods shortly after its integration in an industrial setting?
- **RQ2**: How does MBT perform against conventional testing automation methods over an extended period in an industrial setting?
- **RQ3**: How do user perceptions, attitudes, and expectations of MBT change from initial adoption to prolonged use?

In RQ1, our goal is to assess the effectiveness, efficiency, and maintainability of test cases when initially integrating MBT into an actual industrial setting. Through RQ2, we seek to explore the impact of applying MBT in practice over a period, to determine if MBT truly reduces the testing team's workload and enhances testing efficiency. In RQ3, we aim to understand and analyze user perceptions and attitudes towards the use of MBT in production environments, given that human factors could influence the adoption of new technologies [5].

3.2 Study Design and Execution

Previous studies on MBT typically employ one of the two study methodologies. The first category is surveys that rely on questionnaires or interviews to derive guidelines for MBT implementation, offering qualitative insights and experiences [1]. The second category uses experiments, engaging in comparative analyses between MBT and other test automation methods, focusing on quantitative evaluations through various technical metrics [7, 12, 22, 29, 34]. Prior research has generally favored

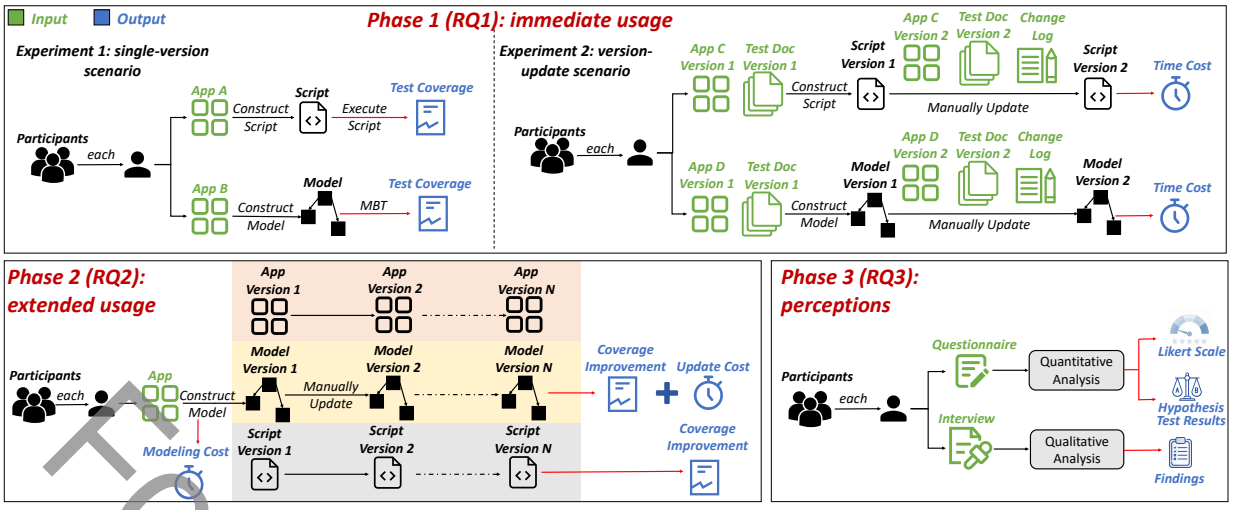

Fig. 3. Overview of the Study Design.

one study methodology over the other. To gather comprehensive data for our in-depth analysis, we utilized a mixed-methods research design, combining controlled experiments, questionnaire surveys, and semi-structured interviews.

Figure 3 provides an overview of the study design. The study is structured into three distinct phases: the immediate comparison phase (Phase 1), which includes two controlled experiments comparing MBT and script writing in both a single software version and a version-update scenario; the extended evaluation phase (Phase 2), which assesses the cumulative effects of the two test methods over a two-month period; and the user perception phase (Phase 3), where participants completed questionnaires to provide feedback on Phases 1 and 2. Additionally, following the two-month Phase 2 experiment, participants were interviewed to share their perceptions of the two methods in terms of effectiveness, efficiency, and maintainability. Collectively, these phases offer a comprehensive evaluation of MBT's comparative effectiveness, efficiency, and stakeholder perceptions.

3.2.1 Phase 1 (RQ1). Like many other companies in the industry, the corporation collaborating with us on this study typically conducts automated testing in two key scenarios. The first scenario is *system testing* for a single version of an application, focusing primarily on the testing effectiveness. The second scenario is centered on *regression testing*, which entails adapting test cases from the base version of an application to its updated version, with an emphasis on efficiency and testing maintainability. Our experiments focus on these two key scenarios; however, it is important to note that the capabilities of MBT extend beyond these scenarios.

To address RQ1 and enable a thorough comparison between the conventional testing method and MBT in a realistic industrial context, we designed two controlled experiments corresponding to the aforementioned testing scenarios. The aim of the first experiment was to mirror the system testing environment while the second experiment was designed to simulate the regression testing scenario. To conduct the experiments, we recruited 12 participants from the corporation and selected six applications currently in use. The participants are all from the development teams of those applications. This ensured a consistent technological background among the participants. We also collected demographic information of each participant. All participants are development team members with over five years of experience but no prior MBT experience. Their attitudes towards technology adoption are detailed in the questionnaire and interview results. Table 1 lists

Table 1. Basic Information of Six Experimental Apps.

Name	App1	App2	App3	App4	App5	App6
LOC	27,346	32,292	18,521	28,078	16,101	17,054

the basic information of the newest versions of six applications. For each application, the table gives the number of lines of code (LOC). For confidentiality reasons dictated by the company's security policies, we have anonymized the names of the applications. These applications were chosen by our collaborators of the company for their diversity, representing a wide range of application types including social, financial, entertaining, and so on, thereby covering a broad array of testing scenarios. We then elaborate on the two experiments:

Experiment 1 is designed to assess the testing effectiveness of MBT compared to conventional script writing, within a constrained time frame. The decision to compare MBT with script-based testing comes from the widespread use of script writing in industrial software testing, including within our collaborator, where it serves as the primary method for test case development. Script writing is widely adopted in the industry due to its flexibility and control over test execution, making it a suitable benchmark for evaluating the potential advantages of adopting MBT. To mitigate bias due to prior familiarity with the applications and ensure a level playing field throughout the study, each of the 12 participants was allocated two applications (a total of 12*2 assignments) with which they had no previous experience or knowledge. Each participant was allocated two hours to conduct comprehensive testing of the applications from scratch, without access to existing testing documents, using both testing methods: MBT and script writing. During the testing, they were provided with the design documents of the assigned applications to prepare for the testing process. For MBT, the participants manually build the models and then the test cases are generated automatically. For script writing, the participants manually write the scripts in Python using the self-made testing framework in our industrial collaborator. Throughout this experiment, we collected data on the line coverage achieved by each participant using the two distinct testing approaches for the applications under test. The average line coverage was then calculated and used as the metric to gauge the testing approaches' effectiveness. We choose line coverage as the metric since our study aims to inspect the industrial practice of automated testing for GUI applications. In such scenarios, an app version can be considered to pass the test only if its testing line coverage meets the pre-defined standard and all failures identified during testing are repaired. Testers hope for the test to meet this coverage standard as quickly as possible to adhere to release time pressures. Moreover, line coverage is a widely used metric in testing automation research because it can reflect the potential to uncover more failures [18]. In MBT, higher testing coverage also correlates with better model quality.

Experiment 2 aims to evaluate the efficiency and maintainability of testing between the newly introduced MBT approach and the existing practice of script writing, within the same test scope. Participants were assigned two different applications from those they worked on in Experiment 1, ensuring a fresh testing context. Also, it was ensured that the participants had no prior experience with those newly assigned applications. Different from Experiment 1, each application had two versions in this experiment, referred to as V1 and V2, accompanied by change logs that detailed the updates from V1 to V2. Furthermore, each version of the applications came with corresponding test case documentation. For the first application assigned to them, participants were tasked with manually writing test scripts for V1 based on its test documentation. Then, they adapted these scripts to meet the testing requirements specified in V2's documentation. For the second application assigned, participants engaged in a comparable procedure but utilized the MBT tool instead. As such,

each of the six applications underwent testing four times (resulting in 6*4 total testing instances): twice using script writing and twice using MBT. Throughout the experiment, we recorded the time spent on each testing task for both versions, and the average time taken was subsequently used as the metric to assess both the efficiency and the maintainability of testing.

Before conducting the above two experiments, all participants were required to attend a ninety-minute training session on MBT. This session provided an overview of MBT's fundamental principles and the specifics of T-IFML, operational guidelines, and a range of modeling examples. The purpose of this training was to ensure participants possessed the necessary foundational knowledge to use the MBT tool. Additionally, instructional materials related to the MBT tool were made available throughout the study, enabling participants to consult these resources as needed. Given that all participants were already proficient in writing test scripts, and regularly engaging in this activity in their daily work, no additional instructions on script writing were offered. After the training, the 12 participants proceeded with the experiment as planned. Each participant worked independently and was not informed of the study's objectives to minimize bias. Furthermore, to mitigate potential learning and order effects that could influence the experimental outcomes, a counterbalancing technique was employed: half of the participants started the experiment by writing scripts, whereas the other half began with MBT.

3.2.2 Phase 2 (RQ2). The investigation of the extended impacts of MBT in industrial environments has received limited attention in previous research. To address this gap, we formulated RQ2 to explore this aspect. Our study involved a thorough evaluation of testing effectiveness and associated costs over a two-month period. During this period, the applications underwent an average of 40 updates per app, with each update undergoing at least one round of testing and some requiring 2-3 iterations. Given the rapid pace of development and frequent testing cycles, this duration provides substantial insights into the practical implications of MBT. From a software development lifecycle perspective, the intensity and frequency of updates and testing over this period align with the conditions typically regarded as extended in dynamic industrial contexts.

Six individuals from the RQ1 participant group, each associated with a different development team responsible for one of six applications, were selected for this phase. In the two experiments of RQ1, participants were assigned applications they were unfamiliar with to ensure an unbiased assessment. This method is effective for evaluating the immediate impacts of MBT, as it eliminates bias from previous knowledge of the applications. However, it does not suit the goal of studying MBT's extended effects in a practical setting, where engineers typically test and maintain the same applications over time. To better align with our objective of assessing the extended implications, we reallocated applications to participants according to their respective development teams, ensuring each engineer from the team responsible for an application was assigned to that specific application. In summary, six participants were tasked with modeling and managing their team's application model for MBT, alongside continuing the conventional test automation practice of script writing. To measure the outcomes quantitatively, we collected data on line coverage and time expenditure for both script writing and MBT methodologies. Additionally, to gain a qualitative understanding of the participants' views, an interview about the participants' perspectives and expectations about MBT was conducted (refer to RQ3 for further details). These discussions sought to uncover the participants' insights on the extended advantages and obstacles of utilizing the MBT tool, thus shedding light on its practical utility and the nuances of incorporating MBT into industrial practices.

3.2.3 Phase 3 (RQ3). To investigate RQ3, we carried out a post-experiment questionnaire survey complemented by an in-person interview. The survey presented 12 participants with questions rated on a 5-level Likert scale [19, 27]. These questions primarily explored the participants' perceptions of:

- Q1: Effectiveness comparison between script writing and MBT.
- Q2: Efficiency comparison between script writing and MBT.
- Q3: Comparison of testing maintainability between script writing and MBT.

Our goal with this questionnaire was to gain a deeper understanding of the participants' perceptions and attitudes towards the adoption of MBT. To determine if the differences in the three metrics between MBT and script writing are statistically significant, we performed hypothesis testing. We used the Wilcoxon signed-rank test for our statistical analysis and Cliff's δ to assess the effect size for each aspect [13].

Following the questionnaire survey, we developed a comprehensive set of interview questions, integrating our research objectives with key insights derived from previous experiments and surveys [1, 22]. This approach enabled a more nuanced investigation into the underlying causes of the observed phenomena. One of the authors conducted in-person interviews with six participants who had been involved in both Phase 1 and 2, ensuring continuity and depth of understanding. Each interview was structured to last approximately 30 minutes, allowing sufficient time for detailed discussions.

The interviews began with discussions on the participants' perceptions of MBT's advantages, encouraging them to elaborate on the factors influencing their views. We then engaged the participants in discussions about the study's preliminary findings, inviting their reflections on how well the results aligned with their expectations and practical experiences. Additionally, participants were asked to share their hands-on experiences with the MBT tool, including its usability, integration into their existing processes, and perceived impact on testing efficiency. To conclude, we sought feedback on potential areas for improvement and asked participants to express their future expectations for MBT, fostering a collaborative exchange of ideas aimed at refining both the tool and its application.

We followed a rigorous process to abstract the interview data. Initially, three authors independently reviewed the transcripts and summarized each participant's responses. In case of disagreements, the different versions of the summaries were sent back to the interviewees for clarification, allowing them to determine which version was most accurate. After the initial summarization, a second round of analysis identified common responses. Semantically equivalent statements from different interviewees were grouped, and the results were documented along with the corresponding numbers of interviewees.

4 Results

This section analyzes the results and presents the findings of the study.

4.1 RQ1: Immediate Effectiveness and Efficiency

We start by examining the results of Experiment 1 within RQ1, which assessed the test outcomes (i.e., line coverage) achieved by testers within a limited time frame using MBT and the script writing approach, as presented in Table 2. This experiment offers a straightforward comparison of the immediate effectiveness of the two testing approaches. The results indicate that, in this scenario, both MBT and script writing have their respective strengths and weaknesses, with MBT showing a slight overall advantage. Script writing demonstrated greater effectiveness for two of the applications, whereas MBT outperformed in the remaining four.

To better understand these results, we conducted a thorough review of the models and scripts used in Experiment 1. For the two applications where script writing yielded better results, it was noted that some behaviors were not fully modeled in the models used in MBT, leading to test coverage gaps. This could be due to testers overlooking certain behaviors or lacking sufficient

Table 2. Comparison of Line Coverage for Script Writing vs. MBT.

App	Script Writing	MBT
App1	23.50%	25.10%
App2	42.80%	49.10%
App3	52.25%	55.15%
App4	19.85%	26.80%
App5	48.30%	34.45%
App6	48.10%	39.30%

Table 3. Comparison of Time Cost (in Minutes) for Script Writing vs. MBT Across App Versions (V1 and V2).

	Script Writing		MBT	
App	V1	V2	V1	V2
App1	40	5	105	30
App2	135	45	68	9
App3	135	12	75	15
App4	35	12	77	7
App5	68	20	51	9
App6	55	20	39	7
Overall	468	114	415	77

proficiency with the tool. In contrast, these behaviors were covered by scripts, likely because engineers are adept at writing scripts for application testing.

In the four applications where MBT showed better performance, some scenarios were still missed despite having more comprehensive models. However, MBT's methodical coverage of modeled behaviors resulted in more comprehensive test cases compared to human-written scripts, even though the models were incomplete and some scenarios were not covered. Overall, in the initial stages of adopting MBT for testing purposes, it has been observed that MBT offers effectiveness comparable to conventional script writing within the same time frame, despite testers having only a basic understanding of the MBT tool and the models still have room for improvement.

Table 3 presents the results of Experiment 2, which primarily evaluates the efficiency and testing maintainability of using MBT versus script-writing-based testing within a immediate context. This experiment emulates a regression testing scenario to mirror real-world industrial testing environments, presenting the time (in minutes) testers dedicated to each method across various application versions. Given that test coverage figures are consistent with those in Table 2 and there were no significant differences between versions V1 and V2 across all six applications for both testing approaches, coverage numbers have been omitted for clarity, as they do not demonstrate a comparative difference between the two methods. Overall, regardless of the version (V1 or V2), the total time required using the MBT approach (415 and 77 minutes, respectively) was less than that for script writing (468 and 114 minutes, respectively). It is worth noting that the testers were new to MBT, while they were already proficient in script writing. This suggests that even in the short term, MBT has the potential to reduce the testing workload compared to script writing.

However, for two of the six applications—App1 and App4—the time required for MBT exceeded that of script writing. Further analysis of all experimental data was conducted to identify the reasons behind these results. App1 and App4 required many test cases with varied inputs, and these applications produced different outcomes depending on the specific input data. Although our MBT tool provides modeling semantics and user-friendly approaches for these scenarios, the participants' lack of familiarity with the tool posed challenges, increasing the time cost. Yet, we observed that once participants became accustomed to the tool, the time required for using the MBT approach significantly decreased when applied to new versions, as highlighted in the next section. For the other four applications, the time needed for MBT was less than that for script writing. In subsequent interviews with testers of these applications, they highlighted that the structured workflow of MBT contributed to its efficiency. They noted that the MBT tool could generate a broad range of potential scenarios and test cases from the model and updating the model is very straightforward. This eliminates the need to manually identify and update all the changes in testing scripts resulting from version updates.

In summary, based on the results of these two experiments, the MBT tool shows comparable effectiveness, efficiency, and maintainability to conventional automation testing approaches in immediate testing scenarios. As testers enhance their modeling skills and refine the models through practical use, we expect them to achieve even greater effectiveness and efficiency in the long term. This will be the focus of our next RQ.

4.2 RQ2: Extended Effectiveness and Efficiency

Before the implementation of MBT at our industrial collaborator, engineers were required to write test case documents for the applications, and then manually convert these into scripts for automated execution. Engineers reported that creating test case documents and scripts typically took an hour or more for a new application version. Additionally, as applications evolved, the test case documents can become too complex to organize, often requiring complete rewrites and escalating the time and effort needed. The shift to MBT marked a significant transformation in the testing process. Testers now develop a model for the application, eliminating the need for conventional test documentation. From this model, the MBT tool automatically generates test cases. Any updates to the application later on necessitate only modifications to the model.

Previous studies [9, 28, 31, 32] have highlighted this advantage of MBT, emphasizing its positive impact on extended software testing, though empirical evidence supporting this claim has been limited. Our RQ2 aims to fill this gap by conducting a comprehensive study on the extended application of MBT in a practical industrial environment. Table 4 shows the testing outcomes for six applications using both MBT and script-writing approaches from June to August 2023. Columns 2 through 5 display the line coverage achieved by each testing approach, while column 6 shows the initial time spent building the MBT model, and column 7 shows the accumulative time spent updating the model for all the updated versions during these two months. The time invested in writing test documents and scripts was difficult to quantify precisely as engineers worked intermittently, and this process often exceeds an hour per version. It is important to mention that these six applications received many updates (nearly one update per day) over this period, with each update necessitating writing test documents and scripts. Due to space constraints and the frequency of updates, we only present the initial and final line coverage in the table.

Initially, MBT outperformed script writing in line coverage for only two applications, with its performance slightly lagging behind script writing for the other four. This is consistent with RQ1 findings, which showed that MBT and script writing perform comparably in the short term. However, over time, MBT's line coverage significantly improved, surpassing script writing across all applications, with an average increase of 13.92% in overall line coverage. This clearly illustrates

Table 4. Line Coverage Improvement and Time Cost of Script Writing vs. MBT.

App	Writing Scripts (2023.6)	Writing Scripts (2023.8)	MBT (2023.6)	MBT (2023.8)	Modeling Cost (mins)	Update Cost (mins)
App1	44.10%	45.60%	38.30%	60.60%	240	120
App2	45.40%	48.10%	45.30%	61.30%	300	80
App3	37.50%	37.90%	53.30%	55.60%	120	60
App4	40.70%	43.90%	38.40%	50.00%	180	80
App5	51.30%	53.80%	42.80%	54.40%	60	80
App6	31.00%	31.40%	43.90%	62.30%	300	100

MBT's extended effectiveness. Additionally, this table shows that script writing's coverage remained relatively stable, whereas MBT's coverage saw substantial gains during the two-month period, indicating its extended effectiveness.

The reason behind this phenomenon lies in two sides. For MBT, the initial model may lack some scenarios and contain errors due to testers' omissions or unfamiliarity with the approach. Over time, the model is continuously refined and more testing scenarios are added. These new scenarios, as we observed, consist of two key aspects: testing new features and functionalities, and covering scenarios that were previously omitted. These refinements will be reflected in the generated test cases by our MBT tool, leading to an increase in test coverage. For script-writing, adding more test cases with significant code coverage improvement requires carefully reviewing and understanding the original test cases and thinking of uncovered scenarios. In an agile environment, this approach is unrealistic and not adopted by the industry. Instead, testers only add test cases for newly introduced features, leaving previously uncovered corner cases unaddressed.

Furthermore, while building the initial model took some effort, the surprising part was that the total time spent on updating models did not surpass 120 minutes for any application, even with the numerous updates throughout the period. Overall, the time required for updates was less than the initial modeling time, except for App5, which necessitated an additional 20 minutes for updates. The reason for this minimal updating time cost is due to the advantages of modeling. Testers can directly represent the behavior of the applications under test through the model, without needing to manually identify existing test scripts and design complex test cases. In MBT, test case generation is fully automated. In our experiments, the generation time varied depending on the model size, ranging from 30 seconds to 2 minutes. This is highly efficient for model updates, as both the generation process and subsequent testing are automated, requiring no manual intervention. Once the initial model, which covers the majority of the application, is constructed, managing it becomes straightforward. Testers can easily add new elements or adjust existing ones using the modeling tool. In contrast, script writing becomes more challenging to manage, especially as the number of scripts increases over time.

To sum up, when looking at the entire MBT process, including both modeling and updating, its cumulative cost was significantly less than that of writing and updating scripts, as our industrial collaborator's experience shows that script writing and updates are extremely time-consuming. This finding supports the extended efficiency and maintainability benefits of using MBT.

4.3 RQ3: Questionnaire Survey and Interviews

After completing the experiments for RQ1 and RQ2, we carried out a survey among the participants to delve into their views on MBT and script-writing. Figure 4 shows the outcomes from a 5-level Likert scale questionnaire, assessing participants' opinions on both approaches across three aspects:

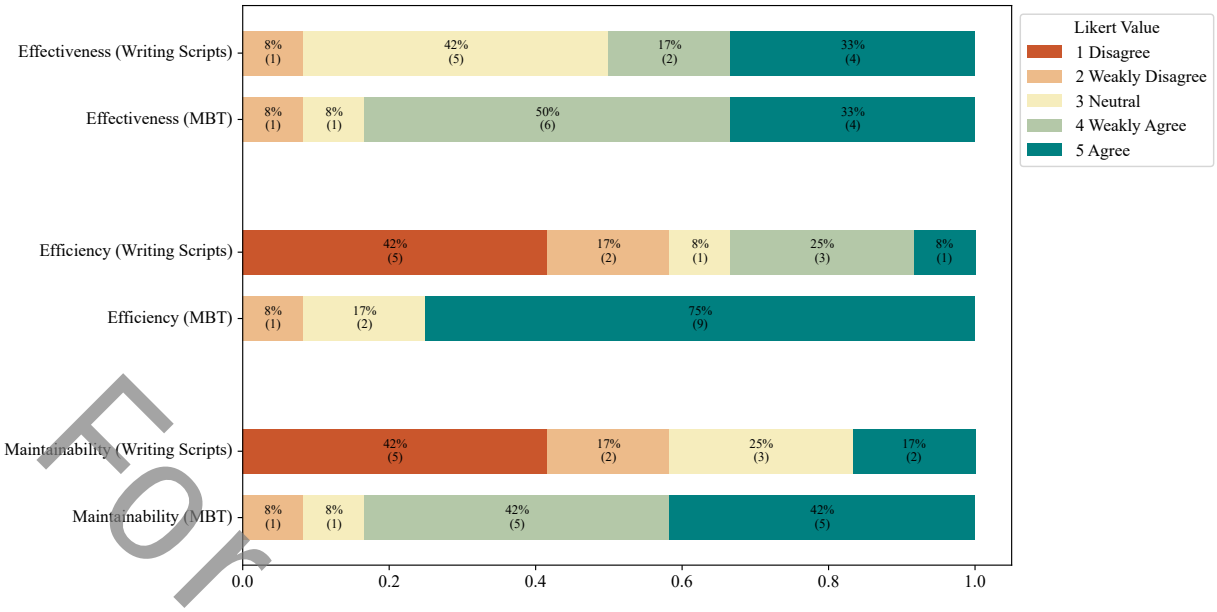

Fig. 4. Results of the Post-Experiment Questionnaire Survey on Participant Feedback.

effectiveness, efficiency, and maintainability. The figure's colored bars indicate how participants rated the two testing approaches according to these aspects, from "disagree" to "agree". Here, "disagree" suggests a negative view on an approach's performance in a certain aspect, whereas "agree" indicates a positive view. The figures and percentages on the bars denote the number and percentage of participants with each viewpoint. For script writing's effectiveness, 42% of participants were neutral, with 50% giving positive feedback. On the other hand, MBT showed a distinct distribution in the weakly agree and neutral categories. Compared to script writing, MBT had a larger share of weakly agreed responses and fewer neutral ones, suggesting a marginally more favorable attitude towards MBT. In terms of efficiency and testing maintainability, script writing received a less positive reception. Only 33% and 17% of participants, respectively, viewed script writing's efficiency and maintainability favorably. Moreover, a significant portion (59%) had negative views on script writing's efficiency, with 42% giving it the lowest score on the Likert scale. In contrast, MBT received a more positive evaluation for both efficiency and maintainability. A majority of participants (75% for efficiency and 84% for maintainability) agreed with positive statements about MBT. Negative views were rare, with only 8% expressing negative opinions for both aspects, and no one rated MBT the lowest on the Likert scale. This feedback underscores the perceived extended benefits and user satisfaction with MBT compared to conventional script writing.

The results of Wilcoxon signed-rank test and Cliff's δ are shown in Table 5. For effectiveness, the p-value was above 0.05, indicating no significant difference in how participants perceive the effectiveness of script writing compared to MBT. However, for efficiency and maintainability, the p-values were below the 0.05 significance threshold, indicating significant differences in perceptions of these aspects. Cliff's δ showed a large effect size for both aspects, underscoring a distinct preference for MBT over script writing regarding efficiency and testing maintainability among participants. This analysis validates MBT's advantage in facilitating more efficient and maintainable testing practices, while views on effectiveness are comparable between the two approaches.

Among all participants who completed the survey, half were involved in both RQ1 and RQ2, while the other half participated only in RQ1. RQ1 assessed the immediate effectiveness and efficiency of

Table 5. Results of the Hypothesis Testing and Effect Size.

Hypothesis	W statistic	p-value	Cliff's δ	Remark
Effectiveness	10.5	0.27	-0.04 (small)	Supported
Efficiency	4	0.01	0.64 (large)	Rejected
Maintainability	4	0.02	0.56 (large)	Rejected

Table 6. Survey Results on Effectiveness, Efficiency, and Maintainability for Immediate vs. Extended Usage Participants.

	Participants of immediate usage		Participants of extended usage	
	Total	Average	Total	Average
Effectiveness	21	3.50	28	4.67
Efficiency	23	3.83	30	5.00
Maintainability	21	3.50	29	4.83

the two testing approaches, whereas RQ2 focused on a longer-term comparison. Hence, we further categorized the survey responses based on whether participants were involved in the extended comparative experiment, dividing them into two groups. Table 6 presents the outcomes: columns 2 and 3 show the total and average Likert scale scores from participants in the immediate comparison, based on three metrics—effectiveness, efficiency, and maintainability. Columns 4 and 5 display the corresponding scores for participants in the extended comparison. The data reveal a clear shift in preference towards MBT among extended participants, with consistently higher scores across all metrics. Notably, the extended participants rated effectiveness, efficiency, and maintainability higher, indicating that MBT's benefits become more apparent with continued use. This suggests that, over time, MBT allows for more streamlined testing processes and better maintainability of test cases. The gradual increase in scores across these metrics underscores the notion that the longer MBT is applied, the more participants appreciate its value. The data further imply that initial challenges or adjustments to MBT in the immediate phase may be mitigated as users become more familiar with the tool, leading to greater extended satisfaction.

In addition to the survey, we conducted interviews with participants involved in both the immediate and extended comparative experiments. The interviewees consistently highlighted MBT's enhanced efficiency, better testing maintainability, and its ability to rapidly generate a wide variety of test cases covering extensive scenarios, which is particularly beneficial for large-scale applications. As participant #2 noted, *"MBT reduces the time needed to create diverse test cases, especially when dealing with complex systems, making it a game-changer for large projects."* Interestingly, all interviewees pointed out that the primary advantage of MBT lies not in the improvement of test coverage, but in the significant reduction of mental effort and the associated time costs. This perspective is shaped by the company's testing objectives, which focus primarily on reaching the required coverage threshold. As participant #4 explained, *"The goal here isn't necessarily higher coverage—it's about reducing the workload and improving efficiency, which MBT handles exceptionally well."* These advantages are largely attributed to the reusability and maintainability of MBT models. Furthermore, the majority (83%) agreed that starting with a basic model and refining it gradually is more practical than attempting to create a detailed model from the outset. This approach mirrors the typical model development process observed with extended MBT usage. Despite acknowledging

the initial learning curve and the time required for modeling, five out of six (83%) interviewees emphasized that these costs are significantly lower than continuing applying traditional testing approaches. These perspectives align with the results of our quantitative analysis. We also examined attitudes toward the MBT tool itself. Features such as graphical modeling and automated alerts for rule violations were frequently mentioned as particularly valuable. These features were progressively integrated during the evolution of the MBT tool. Participant #1 remarked, *"The visual interface and real-time feedback during modeling make it much easier to catch mistakes early, which saves time down the road."* These findings provide valuable insights for other organizations aiming to develop and implement their own MBT tools.

5 Discussion

Our study was conducted in collaboration with practitioners in real-world industrial settings, and the results have been reported and analyzed in the previous section. Throughout the process of applying MBT in these industrial contexts, we gained some insights into how to better implement MBT in industry and identified several related research opportunities. In this section, we discuss these lessons and opportunities, aiming to provide insights for practitioners interested in adopting MBT and for researchers pursuing related studies.

5.1 Implementing MBT in Industry: Lessons Learned

We discovered that implementing MBT in the industry requires advancements in both technical and managerial aspects. Technically, as evidenced by our RQ1 findings, the introduction of MBT demands efforts from testers but yields only comparable testing outcomes initially. Therefore, it is crucial to minimize technical challenges for testers when adopting MBT. This necessitates the selection of appropriate models with semantics closely aligned with developers' common understanding, and sometimes, extending and adapting existing models to enrich their expressiveness. For instance, in our study, we extended IFML to meet the testing requirements of smart TV applications, supporting specific features like remote controller click events and mouse interactions within the Tizen applications. Understanding the unique features of SUTs is essential when developing an MBT tool, yet maintaining simplicity in the model is also vital. Our study emphasizes the need for a balance, advocating for models that are both detailed and manageable.

Additionally, the significance of providing comprehensive training and detailed documentation for testers using MBT is crucial for its successful adoption. As highlighted in the results section, participants' unfamiliarity with MBT had a noticeable impact on their testing strategies and overall effectiveness. Feedback from the interviews further emphasized the need for more robust educational resources, reinforcing the importance of ensuring that the tool is user-friendly. In the early stages of MBT adoption, offering diverse forms of support, such as detailed tutorials, hands-on training sessions, and clear documentation, can significantly help testers quickly gain proficiency in MBT techniques. Adequate training and ongoing guidance are essential for empowering users and fostering confidence in using the tool, especially as testers may initially struggle with its complexities. By managing the tool's complexity through gradual learning processes, organizations can promote smoother transitions and extended adoption. Moreover, the importance of live demonstrations was highlighted by our industry partners, who found these demos invaluable for assessing both the tool's practical utility and its user-friendliness in real-world testing environments.

On the managerial front, our study also provided several lessons. Through RQ2, we found that MBT's advantages are more significantly realized in extended testing. Project managers considering MBT adoption can reference the results of our study, potentially using them as evidence to persuade testers initially reluctant to use MBT. Moreover, the successful application of MBT is inseparable from the continuous refinement of models. Unlike approaches that may be set up once

and left unchanged, MBT demands a proactive and continual process of refinement. Achieving this necessitates managerial support, such as continuous model update management and establishing routine model review and update cycles. Such practices ensure the models' accuracy and usefulness over time. We believe that with ongoing understanding and application of MBT, as demonstrated by the industry participants in our study, testers' perceptions of MBT will improve.

5.2 Implementing MBT in Industry: Research Opportunities

Based on insights from the industry practitioners who participated in this study, we have identified several promising research directions for advancing the implementation of MBT in industrial settings. One recurring issue raised by all six interviewees is the need for tools that can accurately detect semantic errors and provide timely warnings for modeling mistakes. Addressing this challenge could involve exploring advanced automated verification techniques, leveraging metamodels, or applying machine learning algorithms to enhance error detection and correction capabilities. Moreover, since models are central to the MBT process, their continuous refinement in response to changes in the SUT is critical. Currently, the refinement process is largely manual, making it both time-consuming and susceptible to errors. Future research could explore automating this refinement process, offering testers real-time guidance and support to enhance both efficiency and accuracy.

Additionally, while MBT serves as a robust foundation for test case generation, integrating it with other testing methodologies may further enhance the effectiveness of testing strategies. The software testing landscape is dynamic, with new techniques and methodologies constantly emerging. Therefore, MBT tools must remain adaptable and capable of incorporating strategies from other testing paradigms. Future research could explore how MBT can be combined with other testing approaches to optimize test case generation and execution. Furthermore, although MBT models can act as test oracles, aiding in fault detection during the modeling and testing phases, they may not consistently identify more complex functional or non-functional issues. This highlights the need for further research into improving test oracle generation from models, which would significantly bolster the effectiveness of MBT strategies. By focusing on these areas, future studies have the potential to develop broader, more comprehensive testing methods that could enhance software quality assurance across various industries.

6 Threats to Validity

In this section, we discuss potential threats to the validity of our experimental conclusions and the measures taken to mitigate these risks.

6.1 Construct Validity

The primary threats to construct validity in our study arise from the choice of evaluation metrics to assess the effectiveness and efficiency of the testing methodologies. While these metrics are widely used and offer straightforward, quantifiable measures, they may not fully capture all dimensions of testing quality, such as defect detection capability or user satisfaction. However, their direct and objective nature ensures they provide valuable insights into the performance of MBT and conventional testing approaches. To further mitigate this threat, we designed our experiments to focus on key aspects of the testing process that these metrics effectively measure, which, in turn, strengthens the validity of our conclusions.

6.2 Internal Validity

One potential threat to internal validity arises from variability in the participants' familiarity with MBT, which could introduce bias into the experimental results. Differences in prior experience

might affect participants' testing efficiency and effectiveness. To address this, we ensured that none of the participants had formal exposure to MBT before the study, thereby standardizing their starting point in terms of knowledge and skills. Additionally, this study represents the first formal introduction of MBT to the participating company, which further reduces the likelihood of pre-existing biases. While this precaution helps to control for internal validity, we recognize that individual learning curves might still have had some influence. Nevertheless, by carefully controlling participants' backgrounds and ensuring uniform training, we believe the threat is sufficiently mitigated.

Another threat to internal validity comes from the Hawthorne effect [24], which refers to the phenomenon where individuals alter their behavior because they are aware that they are being observed. In our study, several steps can reduce the Hawthorne effect. Participants were informed that the focus was on the testing process rather than their individual performance. They were allowed to work independently, with minimal direct observation. Furthermore, the two-month duration of the study allowed initial biases, such as unfamiliarity with MBT, to diminish over time.

6.3 External Validity

The selection of smart TV applications as the subject of our study introduces a potential threat to external validity, as these applications may not fully represent the diverse range of software systems used in industry. To mitigate this, we utilized the widely adopted cross-platform operating system, Tizen, which supports a broad ecosystem, including smartphones, wearables, IoT devices, and smart TVs. The six active smart TV applications involved in the study cover a variety of use cases, offering a reasonably diversified testing environment. While this approach provides valuable insights, we acknowledge that the generalizability of our findings to other platforms and application types remains a limitation.

Another potential threat to external validity concerns the relatively small participant pool, which may limit the generalizability of our results. We engaged 12 participants from our industrial collaborator, a representative sample within the scope of this study. In future research, we plan to extend our collaboration with additional industry partners and explore MBT's application across a wider variety of software systems and platforms. This broader involvement will help us deepen our understanding of MBT's effectiveness and address the scalability of our findings across different contexts.

7 Related Work

MBT has been a subject of interest for an extended period, with numerous studies dedicated to its exploration [10, 14, 20, 21, 23, 25, 33]. Here, we primarily discuss related empirical studies focused on examining MBT. Among these, particularly in early research, many studies concentrated on MBT's applicability to specific SUTs. Such work mainly investigated whether the MBT approach could be successfully integrated into testing processes, paying less attention to the comparison of its effectiveness against existing testing practices. For instance, Dalal et al. [6] constructed data models for four real-world software products and employed an MBT tool developed by Bellcore to generate input data from these models. Their experimental findings suggested that their MBT approach is applicable to large-scale, data-driven software systems. Sarma et al. [28] assessed two model-based tools against nine criteria in a study on Siemens industrial software in the healthcare domain, sharing the evaluation outcomes.

With the rise of web and mobile applications, researchers also began to explore MBT's application in these domains [7, 30]. Pan et al. [26] proposed a novel MBT approach, Adamant, for automated Android testing based on E-IFML models. They also implemented a path exploration algorithm to facilitate the generation of test cases from these models. Their evaluation, conducted on open-source

Android applications, demonstrates the effectiveness of the MBT approach in terms of code coverage and bug detection capabilities. Entin et al. [9] provided an experience report on implementing MBT in an industrial web development scrum project, noting numerous practical challenges, notably the human factors, such as understanding the modeling language. Takala et al. [31] conducted a case study to outline the experiences and advantages of applying MBT to an Android application, discovering that bugs could be identified during both the modeling and testing phases. Similarly, Karlsson et al. [15] developed an MBT tool for an Android application, demonstrating the successful application of the MBT approach. These studies delve into MBT's application on specific software types but are limited by the lack of comparative analysis with conventional testing methods, affecting their broader applicability. In contrast, our research includes actively-used commercial GUI applications and provides a comprehensive comparison with conventional testing practices, which facilitates a thorough exploration of the effectiveness, efficiency, and testing maintainability of MBT.

In addition, several empirical studies have been conducted to compare MBT with conventional testing methods. Schulze et al. [29] analyzed the performance of script writing versus MBT in testing a web system related to food-borne illnesses. The script writing was executed by a professional tester from industry, while MBT was applied by a researcher. The findings indicated that although MBT required more initial preparation time, it was more effective in identifying functional issues within the software. Marques et al. [22] engaged 12 participants, including both industry professionals and researchers, to test a project for the Federal Police of Brazil using both MBT and script writing. Their conclusion was that both techniques delivered a similar performance in their specific context, with MBT showing greater aptitude for managing complex scenarios. Zafar et al. [34] presented a case study employing GraphWalker, an open-source MBT tool, on an industrial cyber-physical system. They compared the outcomes of MBT against manual testing by industry professionals, revealing that MBT achieved more frequent coverage of requirements than manual test cases. Garousi et al. [12] shared insights from a comparative study between MBT and a proprietary test automation tool, focusing on effectiveness and efficiency metrics. Their evaluation suggested that MBT not only boosted test effectiveness but also improved test-case design practices.

However, these comparative analyses were not performed in a fully authentic industrial setting, and almost none assessed MBT's extended effects. For instance, Schulze et al. [29] implemented the MBT approach in a laboratory environment, and similarly, Garousi et al. [12] conducted comparative conventional testing in lab settings. Unlike these studies, our research was carried out in a genuine industrial production environment, with a particular focus on the extended effects of MBT. This approach, closer to industry realities, led us to new findings. For example, we observed that MBT demands certain initial efforts but only achieves comparable results to conventional methods in the short term, with its advantages becoming more pronounced over a longer duration.

Furthermore, we conducted a questionnaire survey and interviews to examine participants' perceptions of the MBT tool, acknowledging the critical role of human factors in the successful adoption of new technology. We believe insights and feedback from practitioners are crucial for applying MBT, an aspect often overlooked in related work, except by Alégroth et al. [1], who interviewed 17 MBT experts from the software industry. Their interviews yielded 23 best-practice guidelines and underscored the need for more practical use cases and success stories of MBT, reinforcing our motivation for this study.

8 Conclusion

In this paper, we conducted a comprehensive study employing mixed methods to investigate the impact of MBT within a fully industrial setting, in partnership with an international corporation. Through a combined quantitative and qualitative analysis, we uncovered significant findings.

MBT's effectiveness appears comparable to conventional test automation methods upon its initial introduction. However, its benefits become more pronounced over prolonged usage, manifesting enhanced effectiveness, efficiency, and maintainability in testing processes. These results underscore the extended benefits of MBT, supporting its continued application. Moreover, although initial user impressions do not significantly differentiate MBT from conventional testing approaches in terms of effectiveness, these views significantly change over time, with users progressively recognizing MBT's enhanced effectiveness. Drawing from these observations, we derive valuable lessons and research opportunities for industry practitioners and researchers.

9 Data Availability

The data used in our study, including questionnaires, interview materials, and examples of testing scripts and models, are publicly available online[1].

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This research is supported by the National Natural Science Foundation of China under Grant No. 62372227.

References

[1] Emil Alégroth, Kristian Karl, Helena Rosshagen, Tomas Helmfridsson, and Nils Olsson. 2022. Practitioners' Best Practices to Adopt, Use or Abandon Model-Based Testing with Graphical Models for Software-Intensive Systems. *Empirical Softw. Engg.* 27, 5 (sep 2022), 42 pages. https://doi.org/10.1007/s10664-022-10145-2

[2] Andrea Arcuri. 2018. An Experience Report on Applying Software Testing Academic Results in Industry: We Need Usable Automated Test Generation. *Empirical Softw. Engg.* 23, 4 (aug 2018), 1959–1981. https://doi.org/10.1007/s10664-017-9570-9

[3] Marco Brambilla, Andrea Mauri, and Eric Umuhoza. 2014. Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End. In *Mobile Web Information Systems*, Irfan Awan, Muhammad Younas, Xavier Franch, and Carme Quer (Eds.). Springer International Publishing, Cham, 176–191.

[4] Shaoheng Cao, Renyi Chen, Minxue Pan, Wenhua Yang, and Xuandong Li. 2024. Beyond Manual Modeling: Automating GUI Model Generation Using Design Documents. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (Sacramento, CA, USA) *(ASE '24)*. Association for Computing Machinery, New York, NY, USA, 91–103. https://doi.org/10.1145/3691620.3695032

[5] Paul Clarke, Rory V. O'Connor, and Brian Leavy. 2016. A Complexity Theory Viewpoint on the Software Development Process and Situational Context. In *2016 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. 86–90. https://doi.org/10.1145/2904354.2904369

[6] S.R. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C.M. Lott, G.C. Patton, and B.M. Horowitz. 1999. Model-based testing in practice. In *Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002)*. 285–294. https://doi.org/10.1145/302405.302640

[7] Guilherme de Cleva Farto and Andre Takeshi Endo. 2015. Evaluating the Model-Based Testing Approach in the Context of Mobile Applications. *Electron. Notes Theor. Comput. Sci.* 314, C (jun 2015), 3–21. https://doi.org/10.1016/j.entcs.2015.05.002

[8] Vladimir Entin, Mathias Winder, Bo Zhang, and Stephan Christmann. 2011. Combining Model-Based and Capture-Replay Testing Techniques of Graphical User Interfaces: An Industrial Approach. In *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. 572–577. https://doi.org/10.1109/ICSTW.2011.13

[9] Vladimir Entin, Mathias Winder, Bo Zhang, and Stephan Christmann. 2012. Introducing model-based testing in an industrial scrum project. In *2012 7th International Workshop on Automation of Software Test (AST)*. 43–49. https://doi.org/10.1109/IWAST.2012.6228989

[10] Raihana Ferdous, Chia kang Hung, Fitsum Kifetew, Davide Prandi, and Angelo Susi. 2023. EvoMBT: Evolutionary model based testing. *Science of Computer Programming* 227 (2023), 102942. https://doi.org/10.1016/j.scico.2023.102942

[11] Vahid Garousi, Markus Borg, and Markku Oivo. 2020. Practical Relevance of Software Engineering Research: Synthesizing the Community's Voice. *Empirical Softw. Engg.* 25, 3 (may 2020), 1687–1754. https://doi.org/10.1007/s10664-020-09803-0

[1] https://anonymous.4open.science/r/MBT-Study

[12] Vahid Garousi, Alper Buğra Keleş, Yunus Balaman, Zeynep Özdemir Güler, and Andrea Arcuri. 2021. Model-based testing in practice: An experience report from the web applications domain. *Journal of Systems and Software* 180 (2021), 111032. https://doi.org/10.1016/j.jss.2021.111032

[13] Robert Grissom and John Kim. 2005. *Effect Sizes for Research: A Broad Practical Approach.* https://doi.org/10.4324/9781410612915

[14] Katharina Götz, Patric Feldmeier, and Gordon Fraser. 2022. Model-based Testing of Scratch Programs. In *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. 411–421. https://doi.org/10.1109/ICST53961.2022.00047

[15] Stefan Karlsson, Adnan Čaušević, Daniel Sundmark, and Mårten Larsson. 2021. Model-based Automated Testing of Mobile Applications: An Industrial Case Study. In *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 130–137. https://doi.org/10.1109/ICSTW52544.2021.00033

[16] Muhammad Uzair khan, Sidra iftikhar, Muhammad Zohaib Iqbal, and Salman Sherin. 2018. Empirical studies omit reporting necessary details: A systematic literature review of reporting quality in model based testing. *Computer Standards & Interfaces* 55 (2018), 156–170. https://doi.org/10.1016/j.csi.2017.08.002

[17] Anne Kramer and Bruno Legeard. 2016. *Model-Based Testing Essentials - Guide to the ISTQB Certified Model-Based Tester: Foundation Level* (1st ed.). Wiley Publishing.

[18] Yuanhong Lan, Yifei Lu, Zhong Li, Minxue Pan, Wenhua Yang, Tian Zhang, and Xuandong Li. 2024. Deeply Reinforcing Android GUI Testing with Deep Reinforcement Learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024*. ACM, 71:1–71:13. https://doi.org/10.1145/3597503.3623344

[19] Alan Lewis. 1994. Oppenheim, A. (1992). Questionnaire Design, Interviewing and Attitude Measurement, London, Pinter. Pp 303. £14.99 paperback, £39.50 hardback. ISBN 185567 0445 (pb), 185567 0437 (hb). *Journal of Community & Applied Social Psychology* 4, 5 (1994), 371–372. https://doi.org/10.1002/casp.2450040506 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/casp.2450040506

[20] Wen ling Huang, Niklas Krafczyk, and Jan Peleska. 2024. Exhaustive property oriented model-based testing with symbolic finite state machines. *Science of Computer Programming* 231 (2024), 103005. https://doi.org/10.1016/j.scico.2023.103005

[21] Yi Liu, Yuekang Li, Gelei Deng, Yang Liu, Ruiyuan Wan, Runchao Wu, Dandan Ji, Shiheng Xu, and Minli Bao. 2022. Morest: model-based RESTful API testing with execution feedback. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 1406–1417. https://doi.org/10.1145/3510003.3510133

[22] Arthur Marques, Franklin Ramalho, and Wilkerson L. Andrade. 2014. Comparing Model-Based Testing with Traditional Testing Strategies: An Empirical Study. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*. 264–273. https://doi.org/10.1109/ICSTW.2014.29

[23] Guilherme Ricken Mattiello and André Takeshi Endo. 2022. Model-based testing leveraged for automated web tests. *Software Quality Journal* 30, 3 (sep 2022), 621–649. https://doi.org/10.1007/s11219-021-09575-w

[24] Elton Mayo. 1934. The Human Problems of an Industrial Civilization. *Nature* 134, 3380 (01 Aug 1934), 201–201. https://doi.org/10.1038/134201b0

[25] Muhammad Luqman Mohd-Shafie, Wan Mohd Nasir Wan Kadir, Horst Lichter, Muhammad Khatibsyarbini, and Mohd Adham Isa. 2022. Model-based test case generation and prioritization: a systematic literature review. *Software and Systems Modeling* 21, 2 (01 Apr 2022), 717–753. https://doi.org/10.1007/s10270-021-00924-8

[26] Minxue Pan, Yifei Lu, Yu Pei, Tian Zhang, Juan Zhai, and Xuandong Li. 2020. Effective testing of Android apps using extended IFML models. *Journal of Systems and Software* 159 (2020), 110433. https://doi.org/10.1016/j.jss.2019.110433

[27] Simone Romano, Fiorella Zampetti, Maria Teresa Baldassarre, Massimiliano Di Penta, and Giuseppe Scanniello. 2022. Do Static Analysis Tools Affect Software Quality When Using Test-Driven Development?. In *Proceedings of the 16th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement* (Helsinki, Finland) *(ESEM '22)*. Association for Computing Machinery, New York, NY, USA, 80–91. https://doi.org/10.1145/3544902.3546233

[28] Monalisa Sarma, P. V. R. Murthy, Sylvia Jell, and Andreas Ulrich. 2010. Model-Based Testing in Industry: A Case Study with Two MBT Tools. In *Proceedings of the 5th Workshop on Automation of Software Test* (Cape Town, South Africa) *(AST '10)*. Association for Computing Machinery, New York, NY, USA, 87–90. https://doi.org/10.1145/1808266.1808279

[29] Christoph Schulze, Dharmalingam Ganesan, Mikael Lindvall, Rance Cleaveland, and Daniel Goldman. 2014. Assessing Model-Based Testing: An Empirical Study Conducted in Industry. In *Companion Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE Companion 2014)*. Association for Computing Machinery, New York, NY, USA, 135–144. https://doi.org/10.1145/2591062.2591180

[30] Hasan Sözer and Ceren źAhin Gebizli. 2017. Model-Based Testing of Digital TVs: An Industry-as-Laboratory Approach. *Software Quality Journal* 25, 4 (dec 2017), 1185–1202. https://doi.org/10.1007/s11219-016-9321-y

[31] Tommi Takala, Mika Katara, and Julian Harty. 2011. Experiences of System-Level Model-Based GUI Testing of an Android Application. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*. 377–386. https://doi.org/10.1109/ICST.2011.11

[32] Mark Utting and Bruno Legeard. 2006. *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[33] Shuohan Wu, Jianfeng Li, Hao Zhou, Yongsheng Fang, Kaifa Zhao, Haoyu Wang, Chenxiong Qian, and Xiapu Luo. 2023. CydiOS: A Model-Based Testing Framework for iOS Apps. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis* (<conf-loc>, <city>Seattle</city>, <state>WA</state>, <country>USA</country>, </conf-loc>) *(ISSTA 2023)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3597926.3598033

[34] Muhammad Nouman Zafar, Wasif Afzal, Eduard Enoiu, Athanasios Stratis, Aitor Arrieta, and Goiuria Sagardui. 2021. Model-Based Testing in Practice: An Industrial Case Study Using GraphWalker. In *14th Innovations in Software Engineering Conference (Formerly Known as India Software Engineering Conference)* (Bhubaneswar, Odisha, India) *(ISEC 2021)*. Association for Computing Machinery, New York, NY, USA, Article 5, 11 pages. https://doi.org/10.1145/3452383.3452388

Received 2024-09-13; accepted 2025-01-14