A Survey on Software Development, Testing, and Reliability for GUI, Games, Extended Reality, and Embodied Agents

Zhengyang Zhu*†
*School of Software Engineering, Sun Yat-sen University, Zhuhai, China
†Peng Cheng Laboratory, Shenzhen, China
zhuzhy57@mail2.sysu.edu.cn

***Abstract*—Automated software engineering has advanced rapidly across diverse domains, including GUI applications, games, extended reality (XR) systems, and embodied agent environments. This survey reviews representative methods from the perspectives of development, testing, and reliability. Early research on GUI applications focused on model-driven development, event abstraction, and finite-state model-based test generation to ensure functional correctness. With the proliferation of 2D and 3D games, approaches evolved towards system-level scene exploration, search-based testing, multi-agent architectures, and reinforcement learning (RL), improving coverage and navigation in complex virtual worlds. XR applications introduce new challenges, including spatial interactions, long-horizon dependencies, and perceptual test oracles. Methods have been adapted via interaction graph modeling, visual bug detection, field-of-view reasoning, and autonomous NavMesh-guided exploration. Recent embodied agent platforms further emphasize high-fidelity simulation, sensorimotor realism, temporal interaction continuity, and real-time decision reliability. By unifying methodologies, benchmarks, and evaluation objectives across these domains, this survey highlights trends, challenges, and opportunities for automation, cross-domain abstraction, and reliability assurance in spatially and graphically rich software.**

***Index Terms*—Survey, Software Engineering, GUI, Games, Extended Reality, Embodied Agents, Automated Testing, Reliability**

I. Introduction

Software systems with graphical and spatial interactions have evolved rapidly over the past decade, spanning traditional GUI applications, games, extended reality (XR) platforms, and embodied agent environments. Each stage of this evolution has introduced increased complexity in development ecosystems and new requirements for testing and reliability assurance.

A. From GUI to Games

Early GUI software engineering prioritized efficient development frameworks and model-based testing using finite-state abstractions. Automated approaches focused on event sequences and interface correctness, as exemplified by tools such as Stoat [1] and Fastbot2 [2]. As the scale and richness of software grew in 2D and 3D games, manual test specification became insufficient. System-level exploration, search-based testing, multi-agent architectures, and reinforcement learning (RL) emerged to improve coverage and discover complex interaction paths [3, 4].

B. From Games to XR Applications

Building on techniques originally developed for game environments, XR applications further expand both the interaction space and the complexity of automated analysis. While games already feature rich 3D worlds, their interactions remain largely controller-based and occur within constrained camera perspectives. XR applications remove these constraints by introducing six-degrees-of-freedom movement, spatially anchored interactions, and user-dependent viewpoints, thereby transforming many game-testing challenges into harder problems. As a result, methods developed for games—such as search-based exploration, navigation graphs, or RL-driven interaction policies—are increasingly adapted and extended to accommodate XR-specific requirements, including perceptual oracles, spatial alignment quality, and continuous viewpoint-dependent behaviors. Recent approaches incorporate interaction-graph modeling, visual bug detection, field-of-view–based reasoning, and NavMesh-guided autonomous exploration [5–9]. Reliability objectives thus expand beyond crashes or functional errors to evaluating spatial correctness, interaction realism, and immersive usability—all of which are critical but difficult to automate in XR environments.

C. From XR to Embodied Agents

While XR systems primarily support human–environment interaction through virtual reality and (VR) augmented reality (AR) displays, embodied agent environments extend this paradigm by shifting from human-driven actions to autonomous agents that perceive, reason, and act within 3D worlds. Although both domains rely on spatial navigation, viewpoint control, and object manipulation, embodied AI introduces an additional requirement: actions must remain valid not only in simulation but also under real-world physical constraints, creating a fundamental need for sim2real transfer. To support such autonomy, embodied agents depend on high-fidelity simulators such as Habitat [10] and AI2-THOR [11], which provide consistent physics, realistic sensor models, and large-scale task environments. RL-based playgrounds such as Gewu [12] and large language model (LLM)-driven agents like VOYAGER [13] further emphasize continuous perception–action loops, long-horizon behavior, and autonomous skill acquisition. As a result, the boundaries between development, evaluation, and reliability assurance become increasingly blurred, and system correctness depends not only on interaction fidelity—as in XR—but also on the generalization and robustness of policies operating in dynamic, physically grounded environments.

D. Survey Scope and Organization

This survey systematically reviews representative methods across GUI, games, XR, and embodied agents, categorizing them by domain, problem type, methodology, and venue, as summarized in Table I. Figure 1 illustrates the timeline and evolution of these approaches. By comparing methods and objectives across domains, we identify trends in automation, testing coverage, and reliability assurance, and highlight open challenges for future research in spatially and graphically rich software systems.

Graphically rich and spatially embodied software has experienced a clear evolution: from GUI interfaces, to game engines, to XR systems, and recently to embodied intelligence platforms. Each stage expanded both the complexity of development ecosystems and the expectations for testing automation and reliability guarantees. Early GUI software engineering prioritised efficient development frameworks and model-based test generation using FSM-style abstractions, focusing on interface logic correctness and event execution paths. As game development surged, testing research shifted to system-level scene exploration and path search. Graph-based navigation, multi-agent architectures, and search-based testing emerged to increase functional coverage across 2D and 3D gameplay, where manually specifying action sequences became less scalable. The wide adoption of XR introduced fundamentally new challenges: 3D scene state spaces, spatial interactions, long dependency action chains, and perceptual test oracles. Methods evolved towards interaction graph modelling, visual bug detection, FOV-based spatial reasoning, and NavMesh-guided autonomous exploration, with reliability goals expanding to include interaction realism and spatial correctness, rather than only crashes or functional events. The latest wave of embodied agent and spatial-AI development relies heavily on high-fidelity 3D simulators. From a software engineering view, the research objective further extends to environment stability, temporal interaction continuity, sensorimotor input realism, and real-time decision reliability, blurring boundaries between development, testing, and system reliability. This progression reflects a theme shift: oracle definitions evolved from state assertions, to coverage targets, to spatial grounding, and finally to embodied behaviour reliability. The survey that follows organises representative methodologies, benchmarks, and reliability perspectives at each stage, forming a unified understanding of how automation goals, interaction subjects, and verification methods have expanded across domains.

II. Development of GUI, Games, XR, and Embodied Agents

The escalating complexity of graphically rich and spatially embodied software has been met with a parallel evolution in development paradigms and enabling technologies. This evolution is characterized by a shift from manual content creation and scripting towards increasingly automated, data-driven, and intelligent approaches. Key developments include: the adoption of *model-driven engineering* for generating complex game assets; the use of *large-scale RL* to master strategic gameplay; the creation of *high-fidelity 3D simulation platforms* that serve as both training environments and testing grounds for embodied AI; and the emergence of *LLM-powered agents* that automate aspects of the software development lifecycle itself. This section reviews representative frameworks and methodologies that have shaped the development of systems across the GUI, game, XR, and embodied agent domains.

A. Model-Driven Development in Games

Blasco et al. [4] proposed EMoGen, an evolutionary model generation approach for software models in video games within a model-driven engineering context. EMoGen generates high-level software models representing game content such as stages, vehicles, or enemies through an evolutionary cycle of mutation and crossover. Evaluated on the commercial game *Kromaia*, EMoGen produces models with over 1,000 elements in five hours of unattended computation, achieving quality comparable to manually created models. Statistical analysis confirms its effectiveness, demonstrating a significant reduction in developer effort while preserving model fidelity.

B. RL for Games and Embodied Agents

RL has become a foundational technique for developing decision policies in graphically rich games and embodied-agent tasks. *OpenAI Five* [48] demonstrated that large-scale RL with long-horizon reasoning can reach superhuman strategy capability in Dota 2 by enabling self-play training on distributed infrastructure for 10 months, processing millions of frames per second. By defeating the Dota 2 world champion (Team OG), OpenAI Five shows that self-play RL can achieve superhuman performance on highly complex tasks.

Similarly, *Wuji* [3] applied evolutionary optimization combined with deep RL to develop adaptive multi-objective combat policies in online games, demonstrating that RL can autonomously balance survival and exploration objectives during policy learning.

C. Embodied Agent Simulation and Frameworks

Simulated environments are essential for developing and evaluating embodied agents, providing controllable and repeatable settings.

Facebook AI Research proposed Habitat [10], a high-performance platform for research in embodied AI. Habitat supports training embodied agents in photorealistic 3D simulations with high efficiency. It consists of Habitat-Sim, a flexible 3D simulator achieving thousands of frames per second even on a single GPU, and Habitat-API, a modular library for defining tasks, configuring, training, and benchmarking agents. Habitat enables large-scale experiments that were previously

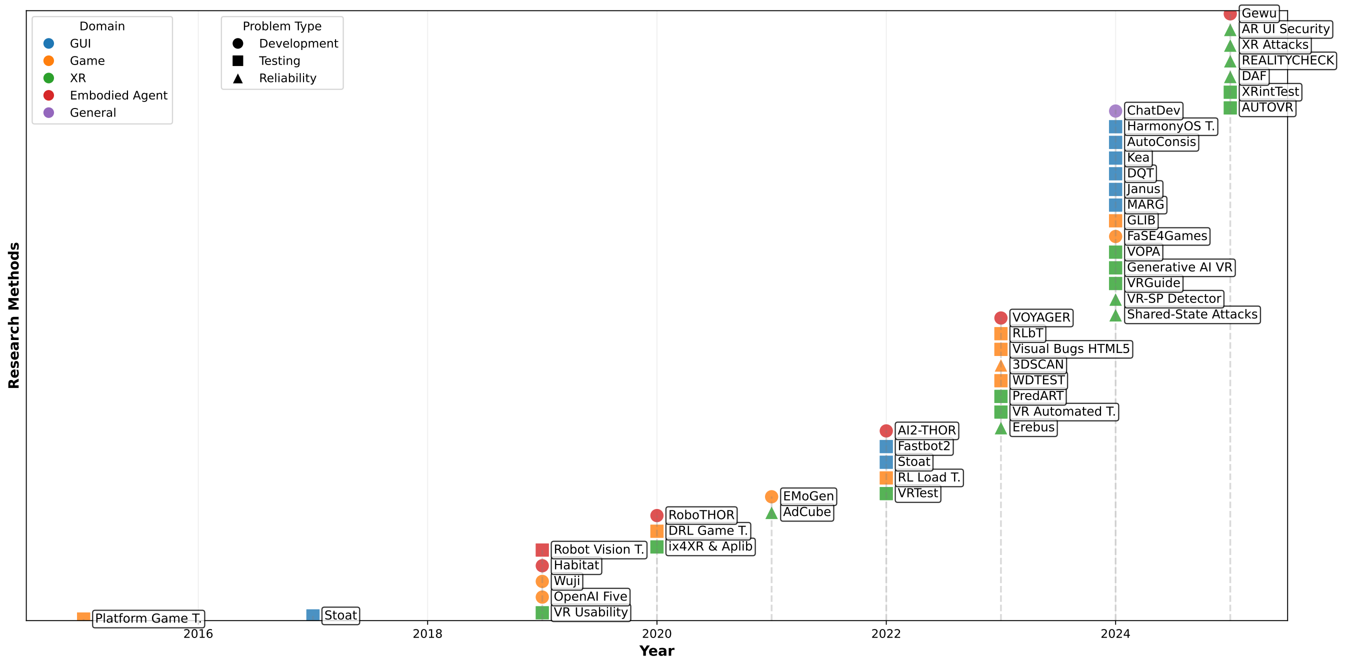

Fig. 1: Timeline of Approaches for GUI, Games, XR, and Embodied Agents Here, "T." indicates Testing / Test.

impractical, including point-goal navigation studies comparing learning-based and SLAM approaches, and cross-dataset generalization experiments with multiple sensor modalities, demonstrating that agents with depth sensors generalize best across datasets.

ROBOTHOR [46] proposed a comprehensive framework of simulated environments to systematically study and address the challenges of simulation-to-real transfer. It further offers a platform that enables researchers worldwide to remotely evaluate their embodied models in corresponding physical environments. AI2-THOR [11] provides a modular and extensible 3D simulation framework where agents navigate and interact with objects in near photo-realistic indoor scenes. From a software engineering perspective, AI2-THOR exemplifies best practices in system design: a scalable Python API for automation, modular scene and object management for code reuse and extensibility, and a controlled environment for testing and validating complex agent behaviors. [13] proposed VOYAGER, an LLM-powered embodied lifelong learning agent for Minecraft that continuously explores the environment, acquires diverse skills, and improves its behavior via iterative self-refinement. VOYAGER manages an automatic curriculum, a skill library of executable behaviors, and iterative prompting with feedback, enabling rapid skill acquisition, temporal composability, and generalization to new tasks without human intervention. Empirical results show it significantly outperforms prior methods in exploration, resource collection, and technology tree progression. GEWU [12] introduces an open-source Unity RL playground built on top of Unity ML-Agents for mobile robot locomotion. It supports one-click training for diverse robot models, multi-mode motion learning, and extreme performance testing, enabling simulation-to-hardware transfer and facilitating robot design optimization and morphological evolution.

D. LLM-Based Software Development

Qian et al. [29] proposed CHATDEV, a chat-powered software development framework where specialized agents, driven by LLMs, collaborate across the design, coding, and testing phases. The framework employs chat chains to guide what agents communicate and communicative dehallucination to guide how they communicate, enabling multi-turn dialogues that produce coherent software artifacts. ChatDev demonstrates that natural language communication can unify multi-agent collaboration, facilitate debugging, and improve consistency across development phases.

III. TESTING OF GUI, GAMES, AND XR

A. Model-Based Testing in Software Engineering

Model-based testing (MBT) [1, 2, 28, 41] is a well-established approach in software engineering that uses abstract models to represent the desired behavior of a testing system. These models are typically FSM, extended finite state machines (EFSM), state transition diagrams, or other formal representations. MBT generates test cases based on the system's model, providing a systematic approach to exploring different scenarios that may not be easily covered through manual testing. By using these models, test engineers can automatically derive action sequences that are executed to validate system behavior under various conditions. [50] proposed a model-based testing approach for automated system-level black-box functional testing of platform games. The

TABLE I: Summary of Approaches for GUI, Games, XR, and Embodied Agents

Year	Paper / Method Name	Domain	Problem Type	Methodology	Venue
2025	REALITYCHECK	XR	Reliability	Enhanced W3C PROV model, NLP and log correlation	USENIX Security
2025	AUTOVR [9]	XR	Reliability, Testing	Automated UI exploration, Dynamic analysis	USENIX Security
2025	DAF [14]	XR	Reliability	SoK-based ontology	USENIX Security
2025	XRintTest [8]	XR	Testing	Automated testing, Interaction graph modeling	ASEW
2025	Gewu [12]	Embodied Agent	Development	RL-based playground, Unity ML-Agents	arXiv
2024–2025	VR-SP Detector [15, 16]	XR	Reliability	Empirical study, Static/Taint analysis	ICSE, TSE
2024	[17]	XR	Reliability	Shared-State Attacks in Multi-User AR	USENIX Security
2024	[18]	XR	Reliability	Demonstrates side-channel attacks in VR	USENIX Security
2024	[19]	XR, GUI	Reliability	Empirical study of UI security properties	USENIX Security
2024	VRGuide [5]	XR	Testing	Computational geometry, Path optimization	ASE
2024	[20]	XR	Testing	Generative AI (GPT-4o), FOV analysis	ASE Workshops
2024	VOPA [7]	XR	Testing, Usability	Screenshot collection, Crowdsourced labeling	ISSTA
2024	[21]	Game	Development, Testing	LLM-based automated unit test generation	FaSE4Games
2024	GLIB [22]	Game, GUI	Testing	Automated GUI testing, Visual bug detection	ESEC/FSE
2024	MARG [23]	GUI	Testing	Multi-Agent RL exploration	ASE
2024	Janus [24]	GUI	Testing	Vision-transformer-based duplicate detection	ICSE
2024	DQT [25]	GUI	Testing	Curiosity-guided Deep Q-Network	ICSE
2024	Kea [26]	GUI	Testing	Property-based testing (PBT)	ASE
2024	AutoConsis [27]	GUI	Testing	Multimodal model (CLIP), Automated analysis	ICSE-SEIP
2024	[28]	GUI	Testing	Model-based testing (PTG), Static analysis	ASE
2024	ChatDev [29]	*General*	Development	Chat-powered software development	arXiv
2023	Erebus [30]	XR	Reliability, Security	Fine-grained access control	SEC
2023	[31]	XR	Testing	Empirical study, Case study	ISSTA
2023	PredART [32]	XR	Testing	Prediction model, Test oracle	ASE
2023	[33]	Game	Testing	Visual bug detection, Automated testing	ASE
2023	RLbT [34]	Game	Testing	RL, Curiosity-driven	ASE
2023	3DSCAN [35]	Game	Reliability	Heuristic-based detection algorithm	USENIX Security
2023	WDTEST [36]	Game, GUI	Testing	Widget detection-based automated testing	ICSE-SEIP
2023	VOYAGER [13]	Embodied Agent	Development	LLM-driven lifelong learning	NeurIPS Workshop
2022	VRTest [6]	XR	Testing	Automated testing, Random/greedy exploration	ICSE Companion
2022	[37]	Game	Testing	RL, Performance testing	ICSE
2022	Stoat [1]	GUI	Testing	Model-based testing, Stochastic model	ICSE
2022	Fastbot2 [2]	GUI	Testing	Model-based testing, RL	ASE
2022	AI2-THOR [11]	Embodied Agent	Development	–	arXiv
2021	AdCube [38]	XR	Reliability, Security	WebVR ad sandboxing with security policies	USENIX Security
2021	EMoGen [4]	Game	Development	Evolutionary model generation	JSS
2020–2022	ix4XR [39–43] & Aplib [44]	XR, Game	Testing	Agent-based testing	A-TEST, EMAS, SSBSE, ICST
2020	[45]	Game	Testing	Deep RL (DRL)	CoG
2020	RoboTHOR [46]	Embodied Agent	Development	Sim-to-real environment	CVPR
2019	[47]	XR	Testing, Usability	Automated usability evaluation, Task trees	TOCHI
2019	OpenAI Five [48]	Game	Development	Large-scale self-play RL on distributed training	arXiv
2019	Wuji [3]	Game	Development, Testing	Evolutionary algorithms, Deep RL	ASE
2019	Habitat [10]	Embodied Agent	Development	High-performance 3D simulation	ICCV
2019	[49]	Embodied Agent	Testing	Robot vision algorithm test platform	IEEE iThings
2017	Stoat [1]	GUI	Testing	Model-based testing, Hybrid analysis	ESEC/FSE
2015	[50]	Game	Testing	Model-based testing, UML state machines	MODELS

Abbreviations: XR (Extended Reality), RL (Reinforcement Learning), DRL (Deep Reinforcement Learning), EFSM (Extended Finite State Machine), PTG (Page Transition Graph), FOV (Field of View), PBT (Property-Based Testing)
Venue abbreviations: ASE: Automated Software Engineering, ICSE: International Conference on Software Engineering, ISSTA: International Symposium on Software Testing and Analysis, TSE: IEEE Transactions on Software Engineering, ESEC/FSE: ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, TOCHI: ACM Transactions on Computer-Human Interaction, CoG: IEEE Conference on Games
MODELS: IEEE/ACM Conference on Model-Driven Engineering Languages and Systems
JSS: Journal of Systems and Software, arXiv: arXiv preprint

work introduces a detailed modeling methodology that leverages domain models to represent game structure and UML state machines for behavioral modeling. Test cases, execution sequences, and oracles are automatically derived from the models. The approach was demonstrated on two case studies: an open-source implementation of the Mario Brothers game and an industrial endless runner game, successfully detecting major faults.

B. Game Testing

Since many VR applications have been developed by game engines, there are several game testing approaches related to our work.

As a code-based data augmentation technique, GLIB [22] can automatically detect game GUI glitches. Macklon et al. [33] proposed an automated visual bug detection method in HTML5 Canvas Games, leveraging internal object representation and multiple similarity metrics to detect visual bugs with high accuracy. Prasetya et al. [39] leveraged graph-based path-finding techniques in automated game navigation and exploration and they proposed a Java-based multi-agent programming framework, IX4XR [39, 40, 42, 43] to facilitate game testing by enabling test agents to interact with the game under test through an interface called *Environment*. Furthermore, they proposed a Belief-Desire-Intention (BDI) library, APLIB [44], to support developing intelligent agents capable of executing complex testing tasks. Ferdous et al. [41] proposed a model-based approach by leveraging EFSM to model game behavior and achieve high coverage and uncover unknown faults in a 3D game. [36] proposed WDTEST, a widget detection-based automated testing framework for mobile games, deployed at NetEase Games. WDTEST constructs the largest GUI dataset for mobile games and performs automated testing using only screenshots as input, without modifying game code. Evaluation shows that WDTEST achieves three times higher unique UI coverage than the commonly used tool Monkey and can be applied to general mobile applications without additional fine-tuning. A survey at NetEase highlights the framework's high compatibility testing performance while pointing out limitations in functionality testing, providing insights into the unique challenges of mobile game QA. [21] proposed an approach for automated unit test generation in game development using LLMs. The method fine-tunes Code Llama to generate unit tests for game-specific scenarios in strongly typed languages like C++ and C#, particularly within the Unity engine. Evaluations show that the approach effectively enhances existing test suites and can generate new tests from natural language descriptions of class contexts and targeted methods.

C. GUI Testing

Stoat [1] generates tests for Android apps by combining dynamic and static analysis to infer a stochastic GUI model, then applying Gibbs-sampling model mutation to guide test generation with increased coverage and sequence diversity. It also injects random system events during execution to expose system-level failures. It was validated at scale, reporting higher code coverage over baseline modeling tools and discovering numerous previously unreported crashes in popular apps. Chen et al. [28] propose a model-based GUI testing approach for HarmonyOS applications with the adoption of the *arkxtest*[51] framework. It introduces a Page Transition Graph (PTG) constructed from the abstract syntax tree (AST) and call graph (CG), and utilizes the *arkxtest*[51] framework for automation. AutoConsis [27] leverages a specially tailored Contrastive Language-Image Pre-training (CLIP) multi-modal model to automatically analyze Android 2D GUI pages. FASTBOT2 [2] leverages a probabilistic model that memorizes key information for testing, which is based on a model-guided testing strategy is another automated model-based GUI testing tool, where the author proposes a probabilistic model that memorizes key information for testing, and a model-based guided testing strategy. KEA [26] is a property-based testing tool for finding functional bugs in Android apps. However, these mobile application testing approaches can only address 2D GUI testing. on Android apps and are incapable of testing 3D VR applications, such as 3D scene exploration. Li et al. proposed JANUS [24], an automated approach for video-based GUI bug report management, focusing on duplicate detection. Janus leverages vision transformers to capture subtle visual and textual patterns on app UI screens, distinguishing similar screens for accurate duplicate identification. Additionally, it employs a video alignment technique with adaptive frame weighting to model typical bug manifestation patterns. Evaluation on 7,290 duplicate detection tasks from 270 video-based bug reports shows that Janus achieves mRR/mAP of 89.8%/84.7%, outperforming prior work by approximately 9% with statistical significance. Fan et al. [23] proposed MARG, the first Multi-Agent RL (MARL) system for automatic web GUI testing. MARG coordinates multiple testing agents to efficiently explore complex websites and share testing experience via centralized or distributed schemes. Empirical evaluation on nine real-world websites shows that MARG significantly increases the number of explored states, detected failures, and unique web states compared to single-agent and state-of-the-art approaches, demonstrating the benefits of cooperative MARL for automated GUI testing.

D. RL-Based Testing

Prior research shows that RL enhances the efficiency of game testing. Tufano et al. [37] employ RL to train an agent to play games in a human-like manner to identify areas that lead to FPS drops. They demonstrated feasibility on three 3D games, including proof-of-concept scenarios with artificially injected performance bugs and an open-source 3D game to evaluate the RL agent's effectiveness in detecting performance bottlenecks. RLBT [34] applies a curiosity-based RL approach to automate game testing by maximizing coverage. Empirical evaluation applies RLBT to a 3D game called Lab Recruits and compares the curiosity-based approach with several alternative baselines. Bergdahl et al. [45] adopts a modular approach, in which RL complements classical test scripting rather than replacing it. Their work expands to include exploit detection and continuous action simulation, such as mouse movements, with a primary focus on navigation rather than combat. *Wuji* [3] leverages evolutionary algorithms, RL, and multi-objective optimization for game testing is a game testing framework that leverages evolutionary algorithms, RL, and multi-objective optimization to perform automated game testing. However, without our framework, RL-based approaches alone are very limited in complex VR scene exploration. Zhu et al. proposed DQT [25], an automated Android GUI testing method built on deep RL to tackle scalability limits of prior Tabular RL approaches. DQT encodes widget structure and semantics as graph embeddings, providing state/action similarity awareness and discrimination for large state–action spaces. A curiosity-guided DQN (Deep Q-Network) is designed to learn interaction testing knowledge from runtime execution and generalize it across states and actions via an enhanced DQN with a specialized DQN architecture. Evaluation on 30 open-source Android apps shows that DQT achieves higher code coverage and fault discovery effectiveness, especially on complex apps. In practice, the approach reproduced and reported faults to developers, receiving 21 confirmed issues, 14 fixes, and 14 patches merged.

E. XR Application Testing

[31] conducts an empirical study on 314 open-source VR applications. The analysis shows that 79% of the evaluated VR projects do not include any automated tests. For the projects that do, the median ratio of functional methods to test methods is lower than that of other types of software projects. [47] presents a fully automated VR application testing approach that does not require users to perform predefined tasks in fixed test settings. Instead, it analyzes recordings of actual usage sessions to generate task trees. These task trees are then examined to identify usability smells, which refer to patterns of user behavior that indicate potential usability issues. PREDART [32] predicts human ratings of virtual object placements, serving as test oracles in AR testing. *VRTest* [6] extracts information from a VR scene and controls the user's camera to explore the scene and interact with virtual objects using specific testing strategies. It includes two built-in strategies: VRMonkey, which uses pure random exploration, and VRGreed, which applies a greedy algorithm to explore interactable objects in VR scenes. *VRGuide* [5] applies a computational geometry technique called Cut Extension to optimize camera routes for covering all interactable objects. VOPA [7] proposed a method for automatic assessment of virtual object placement in AR apps. It instruments real-world AR applications to collect screenshots and metadata, labels them via crowdsourcing, and trains a hybrid neural network to detect imprecise object placements. The approach achieved high accuracy (99.34%) on a test set of 304 screenshots and successfully identified real-world placement errors, enhancing automated usability and quality assessment in AR applications. Qin and Weaver [20] explore Generative AI for field of view analysis in VR testing, showing GPT-4o achieves 63% accuracy in object identification but struggles with organization and localization. However, these approaches cannot address complicated VR interactions and are not generic for different development ecosystems. [8] proposed *XRintTest*, an automated testing framework for Unity-based XR applications. It first constructs an XR User Interaction Graph to model interaction targets and required events, and then automatically explores the XR scene to generate user interactions. Evaluated on XRBench3D, a benchmark with seven XR scenes and 367 distinct 3D interactions, XRintTest achieved 97% coverage of trigger and grab interactions, being 9 times more effective and 5 times more efficient than random exploration, while also detecting runtime exceptions and functional defects.

F. Embodied Agents Testing

[49] proposed a Unity3D-based virtual robot vision algorithm test platform. Their platform simulates underground mine roadway conditions to enable low-cost validation of machine vision algorithms for special robots working in hazardous environments. They conducted experiments on vision enhancement algorithms under darkness and smoke interference. Unlike subsequent XR testing approaches that incorporate spatial interaction signals or hardware-driven input, their work primarily focuses on recreating hazardous visual distributions and algorithmic correctness checking, and does not include closed-loop robotic embodiment or test coverage modelling as a reliability objective.

IV. RELIABILITY

A. XR Application Security and Reliability

[38] proposed AdCube, a security framework for WebVR advertising that detects and confines third-party ad fraud. AdCube identifies novel ad fraud attack methods exploiting the lack of shared canvas support in WebVR. It allows publishers to specify and enforce ad behaviors, effectively mitigating threats while maintaining low page load latency (236 ms) and minimal FPS impact across official WebVR demo sites. 3DSCAN [35] is an open-source tool for 3D model clone detection in mobile games and metaverse applications. It addresses challenges in scalability, precision, and robustness for clone detection, analyzing over 12 million static and 2.5 million animated 3D models extracted from 176K mobile games. The tool identifies likely cloned models—63.03% of static models and 37.07% of animated models—and detects 5,238 mobile games containing potential unauthorized 3D model clones, providing systematic insights into the pervasiveness of 3D model duplication. Kim et al. [30] proposed Erebus, an access control framework for AR platforms. Erebus enforces the principle of least privilege by providing a domain-specific language (DSL) for specifying fine-grained data access for AR applications. It allows users to customize app permissions under specific conditions and supports existing AR apps on Google ARCore. Evaluation demonstrates that Erebus effectively secures AR applications with negligible performance overhead. Yang et al. [18] investigated the privacy risks in VR environments, showing that VR cannot fully protect users from side-channel keystroke inference attacks. They designed attacks in shared VR settings, where an attacker observes another user's avatar hand motion to reconstruct typed content without knowing the keyboard layout or labeled data. User studies across multiple VR scenarios with 15 participants demonstrated that the attacks accurately recover 86%-98% of typed keys and preserve up to 98% of the original meaning, highlighting VR's immersive nature as both strength and vulnerability. Slocum et al. [17] investigated security vulnerabilities in multi-user AR applications that rely on a shared virtual state. They demonstrated attacks that maliciously poison the shared state or exploit false inputs to access restricted hologram augmentations. Evaluations across multiple AR frameworks revealed consistent vulnerabilities despite different implementations. The work also discusses potential mitigation strategies and presents an initial prototype to enhance the security of multi-user AR systems. Cheng et al. [19] conducted an empirical study on UI security in AR platforms. They identified three security-relevant properties: Same Space, Invisibility, and Synthetic Input, and demonstrated their implications via proof-of-concept attacks, including clickjacking and object erasure between AR app components. The study evaluated five AR platforms—ARCore, ARKit, Hololens, Oculus, and

WebXR—showing that all platforms allowed at least three of the proposed attacks, highlighting the need for improved AR UI security and potential defenses inspired by 2D UI security lessons. [15] proposed *VR-SP Detector*, combining static analysis, taint analysis, and privacy-policy semantics to audit VR apps extracted from Oculus Quest 2. It performed a study on 500 apps and found widespread vulnerabilities, privacy leaks, and mismatches between policies and real data collection behaviors, including internal policy contradictions. [16] extended this work by scaling to 900 SideQuest Quest 2 APKs and adding user-review analysis, further validating pervasive vulnerabilities, biometric privacy leaks, and inconsistencies within privacy policies. AUTOVR [9] proposed an automatic framework for dynamic UI and event interaction testing in Unity-based VR applications. Motivated by the lack of headless testing tools for the rapidly growing ecosystem of Meta Quest apps, AUTOVR conducts binary-level analysis to uncover hidden interaction events and their dependencies. Compared with traditional GUI testing tools such as Android Monkey, AUTOVR enables deeper event exploration and achieves substantially higher coverage in detecting sensitive data exposures, thereby highlighting the importance of platform-specific automation for VR software testing. Shoaib et al. [52] proposed *REALITYCHECK*, a provenance-based auditing system designed for analyzing AR/VR attacks. It extends the W3C PROV data model to capture AR/VR-specific entities and causal relationships, leverages natural language processing and feature-based log correlation to extract relations from unstructured logs, and introduces AR/VR-aware execution partitioning to remove irrelevant or false causal data. The system was evaluated on Meta Quest 2 with 25 real-world AR/VR attacks, demonstrating accurate provenance graphs with low runtime overhead. Teymourian et al. [14] proposed the *MR Deception Analysis Framework (DAF)*, a Systematization of Knowledge (SoK) approach to study deception attacks in MR. The framework integrates MR security, information theory, and cognitive models, using the Borden-Kopp deception model to develop a comprehensive ontology of MR attacks. It further derives an information-theoretic model to formalize the impact on information channels and a decision-making model to capture cognitive effects, enabling analysis of MR deception attacks on perception, attention, and decision-making.

V. Conclusion

This survey provides a comprehensive overview of software development, testing, and reliability methodologies across GUI applications, games, XR systems, and embodied agent platforms. We trace the evolution from early model-driven GUI testing and FSM-based abstractions, through system-level exploration and multi-agent frameworks in games, to spatially aware and perception-driven testing in XR environments. Recent advances in embodied agents highlight the importance of high-fidelity simulation, sensorimotor realism, and continuous interaction for ensuring both functional correctness and system reliability. Across these domains, we observe a consistent trend: the scope of testing and reliability assurance expands alongside system complexity, moving from simple state and coverage checks to spatial grounding, embodied behavior validation, and real-time decision robustness. Our synthesis identifies key challenges and opportunities for future research, including the unification of cross-domain testing abstractions, the integration of learning-driven exploration strategies, and the development of reliable, automated test oracles that reflect human-like interaction expectations in immersive and interactive environments. This survey thus lays a foundation for guiding research and practice towards more dependable, efficient, and scalable approaches in modern software engineering for interactive, graphically rich, and embodied systems.

VI. Acknowledgement

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB2704100, the National Natural Science Foundation of China (No. 62032025), the Seed Funding for Collaborative Research Grants of HKBU (with Grant No. RC-SFCRG/23-24/R2/SCI/06), the Major Key Project of Peng Cheng Laboratory under Grant PCL2025AS07.

References

[1] T. Su, G. Meng *et al.*, "Guided, stochastic model-based GUI testing of Android apps," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, p. 245–256.

[2] Z. Lv, C. Peng, Z. Zhang, T. Su, K. Liu, and P. Yang, "Fastbot2: Reusable Automated Model-based GUI Testing for Android Enhanced by Reinforcement Learning," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2023.

[3] Y. Zheng, X. Xie, T. Su *et al.*, "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2019, pp. 772–784.

[4] D. Blasco, J. Font, M. Zamorano, and C. Cetina, "An evolutionary approach for generating software models: The case of kromaia in game software engineering," *Journal of Systems and Software*, vol. 171, p. 110804, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220302089

[5] X. Wang, T. Rafi, and N. Meng, "VRGuide: Efficient Testing of Virtual Reality Scenes via Dynamic Cut Coverage," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2024, p. 951–962.

[6] X. Wang, "VRTest: An Extensible Framework for Automatic Testing of Virtual Reality Scenes," in *2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, 2022, pp. 232–236.

[7] X. Yang, Y. Wang, T. Rafi, D. Liu, X. Wang, and X. Zhang, "Towards automatic oracle prediction for ar testing: Assessing virtual object placement

quality under real-world scenes," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 717–729. [Online]. Available: https://doi.org/10.1145/3650212.3680315

[8] R. Gu and J. M. Rojas, "Xrinttest: An automated framework for user interaction testing in extended reality applications," 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:282688322

[9] J. Y. Kim, C. Zuo, Y. Zhao, and Z. Lin, "Autovr: automated ui exploration for detecting sensitive data flow exposures in virtual reality apps," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25. USA: USENIX Association, 2025.

[10] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied ai research," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:91184540

[11] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," 2022. [Online]. Available: https://arxiv.org/abs/1712.05474

[12] L. Ye, R. Li, X. Hu, J. Li, B. Xing, Y. Peng, and B. Liang, "Unity rl playground: A versatile reinforcement learning framework for mobile robots," 2025. [Online]. Available: https://arxiv.org/abs/2503.05146

[13] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," in *Intrinsically-Motivated and Open-Ended Learning Workshop @NeurIPS2023*, 2023. [Online]. Available: https://openreview.net/forum?id=nfx5IutEed

[14] A. Teymourian, A. M. Webb, T. Gharaibeh, A. Ghildiyal, and I. Baggili, "Sok: Come together – unifying security, information theory, and cognition for a mixed reality deception attack ontology & analysis framework," in *USENIX Security Symposium*. USENIX Association, 2025, long Presentation; Open Access. [Online]. Available: https://www.usenix.org/conference/usenixsecurity25

[15] H. Guo, H.-N. Dai, X. Luo, Z. Zheng, G. Xu, and F. He, "An empirical study on oculus virtual reality applications: Security and privacy perspectives," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024.

[16] H. Guo, H.-N. Dai, X. Luo, G. Xu, F. He, and Z. Zheng, "An empirical study on meta virtual reality applications: Security and privacy perspectives," *IEEE Transactions on Software Engineering*, vol. 51, no. 5, pp. 1437–1454, 2025.

[17] C. Slocum, Y. Zhang, E. Shayegani, P. Zaree, N. Abu-Ghazaleh, and J. Chen, "That doesn't go there: Attacks on shared state in Multi-User augmented reality applications," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 2761–2778. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/slocum

[18] Z. Yang, Z. Sarwar, I. Hwang, R. Bhaskar, B. Y. Zhao, and H. Zheng, "Can virtual reality protect users from keystroke inference attacks?" in *33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024*, ser. SEC '24, D. Balzarotti and W. Xu, Eds. USENIX Association, 2024. [Online]. Available: https://www.usenix.org/conference/usenixsecurity24/presentation/yang-zhuolin

[19] K.-H. Cheng, A. Bhattacharya, M. Lin, J. Lee, A. Kumar, J. F. Tian, T. Kohno, and F. Roesner, "When the user is inside the user interface: An empirical study of ui security properties in augmented reality," in *USENIX Security Symposium*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:263913047

[20] X. Qin and G. Weaver, "Utilizing Generative AI for VR Exploration Testing: A Case Study," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops*, 2024, p. 228–232.

[21] C. Paduraru, A. Stefanescu, and A. Jianu, "Unit test generation using large language models for unity game development," in *Proceedings of the 1st ACM International Workshop on Foundations of Applied Software Engineering for Games*, ser. FaSE4Games 2024. New York, NY, USA: Association for Computing Machinery, 2024, p. 7–13. [Online]. Available: https://doi.org/10.1145/3663532.3664466

[22] K. Chen, Y. Li, Y. Chen, C. Fan, Z. Hu, and W. Yang, "Glib: towards automated test oracle for graphically-rich applications," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, p. 1093–1104.

[23] Y. Fan, S. Wang, Z. Fei, Y. Qin, H. Li, and Y. Liu, "Can cooperative multi-agent reinforcement learning boost automatic web testing? an exploratory study," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 14–26. [Online]. Available: https://doi.org/10.1145/3691620.3694983

[24] Y. Yan, N. Cooper, O. Chaparro, K. Moran, and D. Poshyvanyk, "Semantic gui scene learning and video alignment for detecting duplicate video-based bug reports," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3639163

[25] Y. Lan, Y. Lu, Z. Li, M. Pan, W. Yang, T. Zhang, and X. Li, "Deeply reinforcing android gui testing with deep reinforcement learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3597503.3623344

[26] Y. Xiong, T. Su, J. Wang, J. Sun, G. Pu, and Z. Su, "General and practical property-based testing for android apps," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, p. 53–64.

[27] Y. Hu, H. Jin, X. Wang *et al.*, "AutoConsis: Automatic GUI-driven Data Inconsistency Detection of Mobile Apps," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, 2024, p. 137–146.

[28] Y. Chen, S. Wang, Y. Tao, and Y. Liu, "Model-based GUI Testing For HarmonyOS Apps," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, p. 2411–2414.

[29] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun, "Chatdev: Communicative agents for software development," 2024. [Online]. Available: https://arxiv.org/abs/2307.07924

[30] Y. Kim, S. Goutam, A. Rahmati, and A. Kaufman, "Erebus: access control for augmented reality systems," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, ser. SEC '23. USA: USENIX Association, 2023.

[31] D. E. Rzig, N. Iqbal, I. Attisano, X. Qin, and F. Hassan, "Virtual Reality (VR) Automated Testing in the Wild: A Case Study on Unity-Based VR Applications," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, p. 1269–1281.

[32] T. Rafi, X. Zhang, and X. Wang, "Predart: Towards automatic oracle prediction of object placements in augmented reality testing," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2023.

[33] F. Macklon *et al.*, "Automatically Detecting Visual Bugs in HTML5 Canvas Games," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2023.

[34] R. Ferdous, F. Kifetew, D. Prandi, and A. Susi, "Towards Agent-Based Testing of 3D Games using Reinforcement Learning," in *the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2023.

[35] C. Zuo, C. Wang, and Z. Lin, "A peek into the metaverse: detecting 3d model clones in mobile games," in *Proceedings of the 32nd USENIX Conference on Security Symposium*, ser. SEC '23. USA: USENIX Association, 2023.

[36] X. Wu, J. Ye, K. Chen, X. Xie, Y. Hu, R. Huang, L. Ma, and J. Zhao, "Widget detection-based testing for industrial mobile games," in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, ser. ICSE-SEIP '23, 2023, pp. 173–184.

[37] R. Tufano, S. Scalabrino, L. Pascarella, E. Aghajani, R. Oliveto, and G. Bavota, "Using reinforcement learning for load testing of video games," in *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, 2022, pp. 2303–2314.

[38] H. Lee, J. Lee, D. Kim, S. Jana, I. Shin, and S. Son, "AdCube: WebVR ad fraud and practical confinement of Third-Party ads," in *30th USENIX Security Symposium (USENIX Security 21)*, Aug. 2021, pp. 2543–2560. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/lee-hyunjoo

[39] Prasetya *et al.*, "Navigation and exploration in 3D-game automated play testing," in *Proceedings of the 11th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 2020, p. 3–9.

[40] R. Prada, I. S. W. B. Prasetya, F. Kifetew, F. Dignum, T. E. J. Vos, J. Lander, J.-y. Donnart, A. Kazmierowski, J. Davidson, and P. M. Fernandes, "Agent-based testing of extended reality systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 414–417.

[41] R. Ferdous, F. Kifetew, D. Prandi, I. S. W. B. Prasetya, S. Shirzadehhajimahmood, and A. Susi, "Search-based automated play testing of computer games: A model-based approach," in *Search-Based Software Engineering: 13th International Symposium*, 2021, p. 56–71.

[42] S. Shirzadehhajimahmood *et al.*, "Using an agent-based approach for robust automated testing of computer games," in *Proceedings of the 12th International Workshop on Automating TEST Case Design, Selection, and Evaluation*, 2021, p. 1–8.

[43] I. S. W. B. Prasetya, F. Pastor Ricós *et al.*, "An agent-based approach to automated game testing: an experience report," in *Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation*, 2022.

[44] I. S. W. B. Prasetya, M. Dastani, R. Prada, T. E. J. Vos, F. Dignum, and F. Kifetew, "Aplib: Tactical Agents for Testing Computer Games," in *Engineering Multi-Agent Systems*. Springer, 2020, pp. 21–41.

[45] J. Bergdahl, C. Gordillo, K. Tollmar, and L. Gisslén, "Augmenting automated game testing with deep reinforcement learning," *2020 IEEE Conference on Games (CoG)*, pp. 600–603, 2020.

[46] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi, "Robothor: An open simulation-to-real embodied ai platform," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3161–3171, 2020. [Online]. Available: https://api.semanticscholar.

org/CorpusID:215768690

[47] P. Harms, "Automated usability evaluation of virtual reality applications," *ACM Transactions on Computer-Human Interaction*, vol. 26, no. 3, Apr. 2019.

[48] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. W. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," *ArXiv*, vol. abs/1912.06680, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:209376771

[49] Y. Wang, P. Tian, B. Zheng, Y. Zhou, Y. Li, and X. Wu, "Special robot vision algorithm test platform in virtual reality environment," in *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2019, pp. 416–421.

[50] S. Iftikhar, M. Z. Iqbal, M. U. Khan, and W. Mahmood, "An automated model based testing approach for platform games," in *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, 2015, pp. 426–435.

[51] "Arkxtest," 2022. [Online]. Available: https://gitee.com/openharmony/testfwk_arkxtest

[52] M. Shoaib, A. Suh, and W. U. Hassan, "Principled and automated approach for investigating ar/vr attacks," in *USENIX Security Symposium*, ser. SEC '25, 2025. [Online]. Available: https://api.semanticscholar.org/CorpusID:278982543