Article:

Gu, R., Rojas, J.M. orcid.org/0000-0002-0079-5355 and Shin, D. (2025) Software testing for extended reality applications: a systematic mapping study. Automated Software Engineering, 32 (2). 56. ISSN 0928-8910

https://doi.org/10.1007/s10515-025-00523-7

Automated Software Engineering (2025) 32:56
https://doi.org/10.1007/s10515-025-00523-7

Software testing for extended reality applications: a systematic mapping study

Ruizhen Gu[1] · José Miguel Rojas[1] · Donghwan Shin[1]

Accepted: 21 April 2025

Abstract
Extended Reality (XR) is an emerging technology spanning diverse application domains and offering immersive user experiences. However, its unique characteristics, such as six degrees of freedom interactions, present significant testing challenges distinct from traditional 2D GUI applications, demanding novel testing techniques to build high-quality XR applications. This paper presents the first systematic mapping study on software testing for XR applications. We selected 34 studies focusing on techniques and empirical approaches in XR software testing for detailed examination. The studies are classified and reviewed to address the current research landscape, test facets, and evaluation methodologies in the XR testing domain. Additionally, we provide a repository summarising the mapping study, including datasets and tools referenced in the selected studies, to support future research and practical applications. Our study highlights open challenges in XR testing and proposes actionable future research directions to address the gaps and advance the field of XR software testing.

Keywords Software testing · Extended reality · Systematic mapping

1 Introduction

The global market for Extended Reality (XR) has grown significantly in recent years —estimated at USD 77 bn in 2024— and is expected to continue its rapid expansion to cross USD 3 tn by 2037[1], reflecting the increasing adoption and technological

[1] Market forecast available at: https://www.researchnester.com/reports/extended-reality-market/4863

✉ Ruizhen Gu
rgu10@sheffield.ac.uk

José Miguel Rojas
j.rojas@sheffield.ac.uk

Donghwan Shin
d.shin@sheffield.ac.uk

[1] School of Computer Science, University of Sheffield, Sheffield, UK

Published online: 03 June 2025

maturation of XR across multiple sectors. The industry continues to evolve with major technology companies investing heavily in this space. In late 2024, Google announced Android XR, a dedicated XR operating system built for next-generation computing experiences[2]. The platform is developed in collaboration with Samsung for their forthcoming headset (expected in 2025), which might represent a significant shift in the XR landscape, further accelerating mainstream adoption. While entertainment–particularly video games–remains the most popular application domain for XR technologies (Rodriguez and Wang 2017), various other fields have also benefited from its rapid development, including education (Kavanagh et al. 2017), engineering (Tadeja et al. 2020), military (Lele 2013), and medicine (Kim et al. 2017). This broad spectrum of applications underscores the transformative potential of XR technologies beyond consumer entertainment.

XR is an umbrella term encompassing Augmented, Mixed and Virtual Reality (resp. AR, MR and VR). XR applications (hereafter, XR apps) are software programs designed to run on XR-compatible devices. These apps typically feature virtually organised spaces populated with virtual objects and interactive elements, allowing users to explore scenes and engage with digital content. For instance, *Pokémon Go*[3], a phenomenal AR mobile game, utilises GPS and cameras of mobile devices to overlay virtual content onto real-world locations. More immersive experiences are offered through head-mounted displays (HMDs), such as VR headsets (e.g., PlayStation VR2[4]) and AR headsets (e.g., Apple Vision Pro[5] and Meta Quest 3[6]).

As XR apps become increasingly prevalent across diverse and critical domains and multiple platforms and devices, their development and testing have grown significantly more complex (Andrade et al. 2020). XR apps possess unique characteristics that distinguish them from traditional apps, such as mobile 2D apps. These include real-time responsiveness and complex interactions, enabling users to *select* and *manipulate* virtual objects or *navigate* through virtual environments (Doerner et al. 2022). These differences pose unique challenges for software testing. For example, in the context of generating test sequences, Android apps have finite interaction paths when navigating between different *activities* (i.e., individual screens of an app) (Su et al. 2017). In contrast, XR apps involve virtually infinite interaction possibilities; even a simple task, such as moving towards and interacting with a virtual object, requires accounting for countless variations in interaction sequences (Andrade et al. 2023). These complexities necessitate advanced software testing methods to ensure that XR apps operate reliably and meet user expectations.

Many XR platforms now include simulation capabilities that allow developers to test and debug apps without requiring physical headset usage, such as Meta XR Simulator[7]

2 https://blog.google/products/android/android-xr/

3 https://pokemongolive.com

4 https://playstation.com/ps-vr2

5 https://apple.com/apple-vision-pro

6 https://meta.com/gb/quest/quest-3

7 https://developers.meta.com/horizon/documentation/unity/xrsim-intro

and Unity XR Device Simulator[8]. For instance, Meta XR Simulator supports Meta Quest app development by enabling keyboard, mouse, or game controller simulation of XR interactions. The simulator also features a valuable *record and replay* function that captures input sequence and verifies consistent behaviour across executions. While record-and-replay is a common testing approach for GUI apps that simplifies the automation of complex usage scenarios (Hu et al. 2015; Modarressi et al. 2024), it has limitations, such as poor maintainability, where the captured test frequently breaks when the app's UI changes, requiring substantial manual updates to remain effective (Lam et al. 2017).

Unlike traditional software, which benefits from well-established surveys covering various testing practices (Zein et al. 2016; Garousi et al. 2013), to the best of our knowledge, there are currently no available comprehensive systematic review studies dedicated to XR software testing. Critical aspects, such as testing practices, tools, frameworks, and general testing guidelines, remain largely unexplored.

To address this gap, we present a systematic mapping study on XR software testing. Systematic mapping is a methodology designed to survey the literature, provide a comprehensive overview of a topic, identify research gaps, and offer insights into future research directions. By carefully following the guidelines proposed by Petersen et al. (2015), we selected a total of 34 primary studies as the subjects for this mapping (see Appendix Afor the details of the studies). We systematically classified and extracted data from these studies to investigate the current research status in XR software testing, explore key testing facets (e.g., activities and objectives), and examine the evaluation methodologies used. To facilitate future research, we compile and present the tools and datasets used in the studies. Finally, we identify the limitations and challenges in XR software testing and highlight potential avenues for advancing the field.

The main contributions of this systematic mapping study are as follows:

- We provide an in-depth survey of the current software testing methods for XR apps, shedding light on the state-of-the-art in this emerging domain. The data extraction template used to derive these findings is included as part of the study.
- We compile a repository of existing tools and datasets used in XR software testing to support future research in the field. The repository, along with the data extraction results, is publicly available at: https://sites.google.com/view/xr-testing.
- We identify critical challenges in XR software testing and outline potential research directions to address these challenges and advance the field.

This paper is structured as follows. Section 2 presents the background of this work, key definitions and a primer on XR user interaction. Section 3 discusses the motivation behind this work and summarises relevant related studies. Section 4 details our methodology, including the process for searching and selecting relevant literature. Section 5 presents the results of our study and answers to our research questions. Section 6 discusses the findings and explores their implications for the field of XR software testing. Section 7 summarises the key contributions and concludes the paper.

[8] https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/xr-device-simulator-overview.html

2 Background

This section provides background on Extended Reality (XR), covering key concepts and terminology across various immersive technologies that form the foundation of this mapping study. Table 1 lists abbreviations frequently used throughout this paper. We introduce the nature of XR applications and their user interaction models, followed by relevant software testing concepts–particularly focusing on automated testing, test automation, and GUI testing approaches.

2.1 Extended reality

Over recent years, the development of virtual technologies, such as VR and AR, has grown rapidly. These advancements allow users to immersively interact with virtual objects and virtual environments with specific devices, such as HMDs and controllers.

Figure 1 illustrates the differences between XR technologies using the reality-virtuality continuum introduced by Milgram et al. (1994). The continuum spans from fully real environments (reality, on the left) to entirely virtual ones (virtual reality, on the right). The proportion of real versus virtual elements shifts along the continuum: reality diminishes while virtuality increases, and vice versa. AR, MR, and VR represent distinct forms of XR across this spectrum.

Augmented Reality (AR) is positioned near the reality end of the continuum; it overlays virtual objects onto the real world in real time, allowing users to interact with both. A prominent example is the mobile game *Pokémon Go* (Fig. 2a), where virtual creatures and widgets are superimposed onto real-world environments.

Mixed Reality (MR) bridges AR and VR by blending real and virtual environments, enabling real-time interaction between physical and digital elements. Virtual objects in MR behave as if they existed in the real world, offering enhanced functionality and immersion. For instance, car designers can use MR to manipulate 3D models of car components, refining designs with seamless interaction between real and virtual

Table 1 Summary of frequent abbreviations

Abbrev.	Definition
XR	Extended reality, an umbrella term encompassing augmented (AR), mixed (MR), and virtual (VR) reality technologies.
VR	Virtual reality, a computer-generated simulation that immerses users in a virtual environment using 3D displays and motion tracking.
AR	Augmented reality, a technology overlaying digital content onto the real-world visual environment.
MR	Mixed reality, a technology blending real and virtual worlds, allowing physical and digital objects to interact.
DOF	Degree of freedom, the number of independent ways an object can move or rotate in three-dimensional space.
HMD	Head-mounted display, a wearable display device positioned in front of the user's eyes to provide immersive visual experiences.

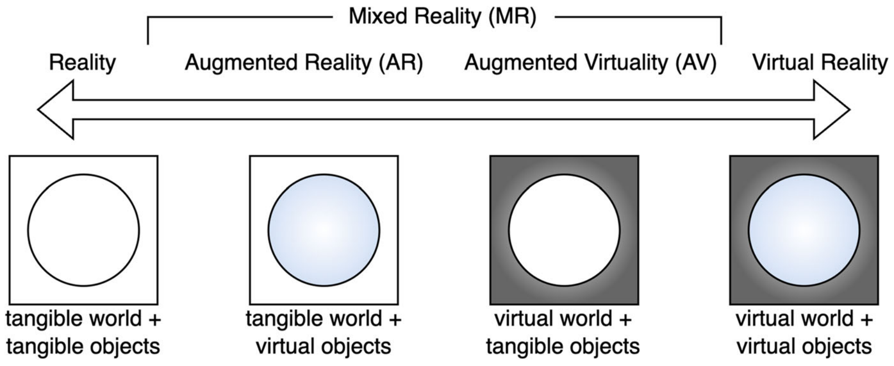


Fig. 1 Reality-virtuality continuum

elements (Fig. 2b [9]). This capability goes beyond enhancing real-world experiences with added information by allowing deeper integration and interaction between realms.

As XR technologies continue to evolve, the distinction between AR and MR has become blurred, with the terms AR and MR often used interchangeably in both industrial and academic contexts. Doerner et al. (2022). For clarity throughout this paper, we maintain the distinction between these two technologies based on the definitions provided above, with AR focusing on overlaying information and MR enabling deeper integration between real and virtual elements.

Virtual Reality (VR) is located at the virtuality end of the continuum, VR immerses users entirely in a digital environment, blocking out the real world. For instance, Fig. 2c shows *Resident Evil 4 VR Mode*[10], a VR video game, where players perform actions like shooting and reloading within a fully virtual setting.

2.1.1 XR applications

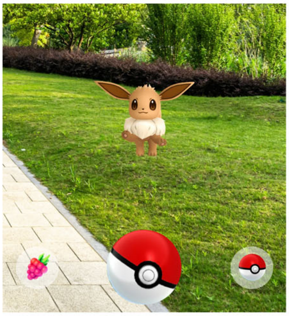
Most XR apps are developed using 3D engines and platforms, like Unity[11] and Unreal Engine[12] (Roberts 2023), which support deployment on various platforms, such as Android, iOS and the web (Scheibmeir and Malaiya 2019; Qiao et al. 2019). A typical XR app consists of interconnected *scenes*, analogous to *activities* in Android apps, each representing a unique virtual environment. Using Unity as an example, scenes are composed of *GameObjects* and *components*. GameObjects are the graphic elements that users can interact with, while components provide functionalities to GameObjects (e.g., animation, video playback) (Technologies 2024). The hierarchical structure of XR scenes, including object relationships and properties (e.g., behaviours, appearances), is managed using specialised data structures called *scene graphs* (Walsh 2022).

9 https://newsroom.porsche.com/en/2024/innovation/porsche-mixed-reality-workshop-augmented-reality-34998.html

10 https://store.playstation.com/en-gb/product/EP0102-PPSA07412_00-RE4RDLC000000028

11 https://unity.com/

12 https://www.unrealengine.com/



(a) AR Example

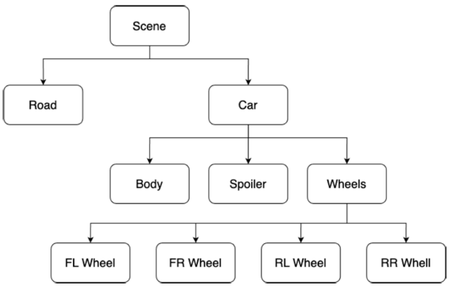
(b) MR Example

(c) VR Example

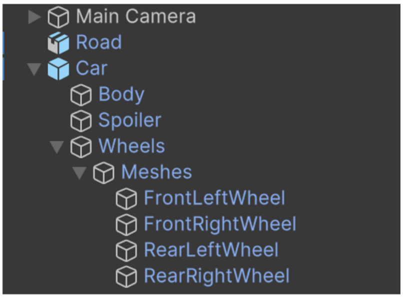
Fig. 2 Examples of AR, MR, and VR scenes

Although XR apps can be built with different frameworks and languages, this core structure is consistent across platforms.

Figure 3a depicts the scene graph of an XR environment, including objects such as a *Road* and a *Car*. The *Car* object is further divided into sub-objects like *Body*, *Spoiler*, and *Wheels*. Figure 3b presents the corresponding XR scene in the Unity Editor, showcasing the hierarchical relationships among these objects.



(a) Unity Scene Structure



(b) Scene Graph

Fig. 3 (a) Scene Graph (b) Unity Scene Structure

2.1.2 Interaction with XR applications

In XR apps, user interaction typically involves three main tasks (Kim et al. 2020; Doerner et al. 2022): (1) *navigation:* controlling the user's position and viewing direction within the virtual environment; (2) *selection:* choosing a point, area, volume, or specific virtual object; (3) *manipulation:* modifying the parameters of virtual objects, such as their location, orientation, or size. Although these tasks are conceptually similar to those in traditional 2D graphical user interfaces, their execution in XR is significantly more complex (Emery et al. 2001; Doerner et al. 2022).

This increased complexity arises primarily from the degrees of freedom (DOF) involved-the number of ways an object can move in space. In 2D interfaces, interactions typically involve rigid bodies with three DOF: two translations (horizontal and vertical) and one rotation. In contrast, 3D objects in XR apps operate with six DOF (*6DOF*), encompassing three translational movements (forward/backward, up/down, left/right) and three rotational movements (yaw, pitch and roll). The additional DOF in XR interactions introduces multiple layers of complexity, demanding more sophisticated interaction techniques and testing methodologies.

The choice of input device is a critical factor in enabling effective interaction within XR environments. Unlike traditional 2D apps, which rely on menus, buttons, and toolbars, XR apps often require specialised hardware to support their unique interaction paradigms. While mobile and web-based XR apps typically run on conventional devices like smartphones and web browsers, delivering more immersive XR experiences usually demands dedicated devices, such as HMDs, These devices are specifically designed to handle complex and dynamic interactions in XR apps, including 6DOF tracking. This capability enables precise mapping of the user's physical actions, such as movement, rotation, and gestures, into immersive environments, allowing more natural and intuitive interactions.

In a virtual environment, user inputs are processed in real-time, where even slight variations can significantly alter scene behaviour and the input sequence required to complete a task (Andrade et al. 2023). This dynamic nature makes reproducing exact input sequences for task replication particularly challenging. In contrast, traditional GUI software, such as 2D mobile applications, can often be modelled as finite state machines (FSMs) (Su et al. 2017), where input events required to reach a specific state are finite and reproducible. This allows for controlled and predictable interactions.

However, XR apps rely on 6DOF interactions and real-world context, introducing unpredictability. For instance, an XR app may present varying virtual content depending on the user's current location or physical surroundings, making it far more complex to test and replicate specific input sequences compared to traditional 2D GUI apps.

Similarly, although XR apps and 3D video games share common foundations, such as development with the same 3D engines (Bouvier et al. 2008), they differ significantly in interaction mechanisms and real-world integration. This systematic mapping study distinguishes XR apps from 3D video games, acknowledging their shared technological roots but unique user experiences and testing challenges.

2.2 Software testing

Software testing is a practical engineering activity in software development, aimed at ensuring the quality of a software system by evaluating the system under test (SUT) (Ammann et al. 2008).

A central element of software testing is the *test case*, which specifies the conditions for executing the SUT in a certain way. Test cases typically include inputs, execution conditions, and expected results, known as *oracles*, to validate the software's behaviour (Washizaki 2024; Barr et al. 20215) and detect faults.

Testing spans multiple levels, each with distinct objectives: (1)*unit testing* verifies individual components, such as methods or classes, in isolation; (2) *integration testing* examines interactions between components, such as method calls across modules; (3) *system testing* assesses overall behaviour, including non-functional requirements like security and usability. These levels ensure comprehensive evaluation, targeting specific aspects of the system's design and functionality.

Manual testing is time-consuming and resource-intensive, making it impractical for exhaustive testing in large programs. While human testers are indispensable for tasks requiring creativity or domain knowledge, automated testing is increasingly relied upon to streamline repetitive tasks and enhance test coverage.

2.2.1 Automated testing and test automation

Automated testing and *test automation* are related terms in the testing domain. We acknowledge these terms might have diverse definitions across academia and industry. For clarity and consistency in this paper, we define automated testing as the automation of both test *generation* and *execution*, while test automation refers solely to the automation of test execution (e.g., driven by manually created test data).

Automated testing reduces the reliance on manual effort by automating the creation and execution of test cases. This approach improves efficiency, consistency and thoroughness in testing, particularly for complex and large-scale systems. Test oracles are an essential part of automated testing and generating accurate and robust oracles is a challenging problem (Molina et al. 2025).

On the other hand, test automation often relies on manually crafted test data, involving script-based testing frameworks. For instance, tools like Espresso and UIAutomator are scripted-based testing frameworks for Android apps, offering intuitive GUI testing approaches for developers (Gu and Rojas 2023).

Complex domains like GUI apps can significantly benefit from automated testing and test automation, as these approaches can systematically simulate real-world interactions and validate expected outcomes.

By addressing the challenges of scalability and repeatability, automated testing and test automation have become essential in modern software development, complementing manual efforts to ensure comprehensive quality assurance.

2.2.2 GUI testing

System testing is crucial for GUI apps, complementing unit testing by focusing on user interactions to ensure the software meets requirements and quality standards.

In 2D GUI apps such as Android apps, system testing typically treats the apps as a black box, interacting with the GUI widgets to validate functionality (Kong et al. 2019). Test automation for these apps can be classified into three generations based on the abstraction level of GUI elements (Ardito et al. 2019): (1) *coordinate-based*: interactions rely on exact screen coordinates; (2) *layout-based*: GUI elements are identified by properties like unique IDs; (3) *image recognition-based*: components are identified through image matching. On the other hand, test generation tools are developed to automatically generate the test cases and execute them within the SUT. They typically create interaction sequences like button clicks or text input to systematically explore the SUT and validate functionality against expected behaviours.

Testing 3D GUIs, such as those found in video games and XR apps, presents significantly greater challenges due to their fine-grained interactivity and complex spatial relationships, which renders automated testing particularly difficult (Politowski et al. 2022) Emerging methods leveraging advanced techniques–such as evolutionary algorithms and reinforcement learning–aim to systematically explore the 6DOF spaces. However, substantial challenges remain, such as handling flexible camera movements to adjust the user's point of view (Zheng et al. 2019).

3 Related work and motivation

To explore related work on XR software testing, we conducted a preliminary literature search. During the process, we identified a few secondary studies, such as systematic mapping studies and literature reviews, vaguely related to testing XR applications. For example, Börsting et al. (2022) conducted an informal review of software engineering techniques for AR apps, analysing their applicability across various engineering phases, including requirement engineering, implementation and testing. With regard to testing, the study emphasised the importance of interaction testing and test automation for AR user interfaces. However, it noted significant challenges, such as the lack of formal definitions for AR-specific interactions and the need for tailored testing approaches for unique AR components (e.g., animations, transformations). These challenges pose significant obstacles to achieving effective automated interaction testing. Additionally, the study discussed the prevalent reliance on user-based usability testing in AR, which often involves labour-intensive user studies. This reliance underscores the need for more reproducible and efficient testing approaches to reduce manual effort and improve scalability. While this study offered a broad overview of software engineering for AR, our work specifically focuses on the unique challenges and methodologies of software testing for XR apps (i.e., including but not limited to AR).

Kuri et al. (2021) conducted a mapping study focusing on software quality metrics for validating VR products (e.g., code quality, audio quality, quality of experience) rather than software testing techniques. The study found that the existing metrics are primarily tailored to specific app types, such as educational apps, making them less applicable to other domains like manufacturing for instance. Researchers tend to develop custom quality metrics and evaluation methodologies, highlighting the need for a general framework that assesses VR app quality across various dimensions (e.g., code, video, audio). With the majority of existing metrics focused on the quality of experience and relying on manual evaluation, the study highlights the need for automated, objective methods or metrics to assess software quality.

3.1 Usability of XR applications

While our preliminary literature search did not uncover any systematic review dedicated explicitly to XR software testing, we did find several studies evaluating XR system's *usability*, i.e., the ease of use of specific software systems (Hertzum 2020).

Ramaseri et al. (2019) reviewed usability and performance evaluation in VR systems. They identified key usability issues including *health and safety issues*, *social issues* and *sensory constraints*. The study also identified usability evaluation methods, such as cognitive evaluation (Brown-Johnson et al. 2015), user analysis (Barbieri et al. 2017), and group testing (Chen et al. 2013).

Dey et al. (2018) conducted a systematic review of AR usability studies from 2005 to 2014. They analysed 369 user studies across various application domains such as education, entertainment, and industry. The most common data collection method was questionnaires, resulting in subjective ratings being the most widely used measure. Kim et al. (2020) reviewed VR systems from a human-computer interaction (HCI) perspective. The findings aligned with those of Dey et al. (2018), highlighting *subjective measures* as the dominant approach for evaluating VR/AR usability.

Both Ramaseri et al. (2019) and Kim et al. (2020) identified *cybersickness* as a significant usability issue in XR systems. Cybersickness, a form of visually-induced motion sickness experienced in immersive environments, manifests through symptoms such as nausea, disorientation and headaches (Davis et al. 2014). Various factors may cause cybersickness regarding individuals (e.g., illness, posture), devices (e.g., lag, calibration), and tasks (e.g., control, duration) (Davis et al. 2014). Studies aiming to comprehensively assess cybersickness by employing subjective and/or objective measures exist. Subjective measures, e.g., the Simulator Sickness Questionnaire (SSQ) (Robert et al. 1993), evaluate participants' self-reported symptoms. Objective measures, in contrast, primarily involve real-time physiological data collection such as heart rate variability (HRV) or eye tracking while participants perform specific tasks (Kamińska et al. (2022); Qu et al. (2022); Kundu et al. (2023)).

Yang et al. (2022) conducted a systematic review focusing on the use of machine learning (ML) techniques to study cybersickness. The review examined 26 studies that utilised ML approaches with biometric and neuro-physiological signals, such as electroencephalogram (EEG) and electrocardiogram (ECG) data obtained from wearable devices, for the automated detection of cybersickness.

These studies emphasise the unique usability challenges posed by XR systems compared to traditional software. Testing user interactions in XR is essential, especially because human behaviour in these environments is highly complex and cannot be mathematically modelled to guarantee predictable outcomes (Doerner et al. 2022).

While automated approaches show potential in addressing usability issues like cybersickness detection, they largely depend on user involvement. This reliance on manual testing or *live* user data collection is time-consuming and costly.

Although these findings highlight the importance of *user-centric* evaluations, they also expose a gap in exploring *software-centric* testing approaches. Unlike user-centric methods, software-centric testing can detect failures earlier in the development process and offer more efficient, systematic, and automated testing capabilities. Our mapping study seeks to bridge this gap by examining studies that address usability issues from a software-centric perspective.

3.2 Motivation

There is a noticeable gap in the literature regarding a comprehensive overview of testing practices for XR applications. While recent research on VR app testing highlights the scarcity of literature on software engineering practices specifically for the VR domain (Andrade et al. 2023), this observation extends to the broader XR domain, which remains significantly underexplored. The gap underscores the need for a formal and in-depth mapping study to analyse existing evidence on XR app testing challenges and techniques, identify research gaps, and suggest further research directions, potentially including systematic literature reviews on specific aspects of XR testing.

Most secondary studies related to XR software testing have primarily focused on usability, a trend that aligns with the findings of Kuri et al. (2021). While usability is important for XR experiences, this limited focus has a significant gap in understanding the broader landscape of XR software testing methodologies. To fill this gap, we conduct this systematic mapping study to provide a comprehensive overview of software testing methodologies for XR apps. Our focus is on techniques and frameworks that prioritise *software requirements*, specifically addressing XR software testing challenges.

Our study adopts an inclusive approach by incorporating empirical studies that, while not directly proposing testing methods, offer valuable insights or theories beneficial to testing. For example, studies analysing common bug types in XR apps provide foundational knowledge that can inform the development of testing strategies. This broader inclusion ensures a more holistic understanding of XR software testing.

4 Mapping study

This systematic mapping study follows the guidelines proposed by Petersen et al. (2015) and is inspired by other systematic mapping studies (Zein et al. 2016; Zhang et al. 2023). As shown in Fig. 4, our mapping process consists of three stages: (1) planning, where the research questions and the scope of the literature search are formulated, (2) conducting, where the authors specify a search strategy, search, and select primary studies, then apply classification and data extraction processes to them subsequently, (3) reporting the mapping, presenting the outcomes of the study, with complete details of primary studies and extracted data available in the appendix and repository.

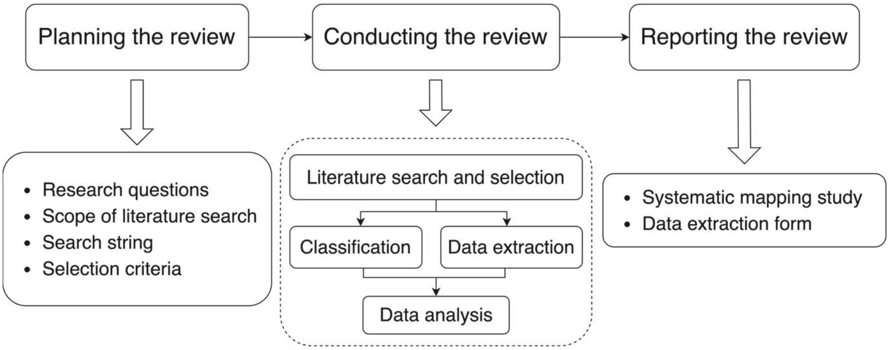


Fig. 4 Process of the mapping study

4.1 Planning the mapping

4.1.1 Research questions

This study aims to develop a comprehensive classification scheme by analysing relevant evidence and insights from the existing literature on software testing for XR applications. The scope extends beyond the studies that introduce novel testing techniques for XR apps, encompassing a broader range of research, including empirical studies that provide valuable information for XR software testing (e.g., analysing common bug types within XR software). Moreover, the study seeks to identify research gaps and challenges and outline future research directions. We therefore formulate the following research questions (RQs):

RQ1: ***What is the current status of XR application testing research?*** This question provides an overview of the current landscape in XR software testing research. It will explore general aspects such as the number of publications over recent years, major publication venues, and common research types. Additionally, we will investigate the most discussed and emerging topics within XR testing research and which XR technologies (e.g., VR, AR) are the primary focus of current research.

RQ2: ***What are the test facets involved in XR applications?*** This research question aims to provide a comprehensive overview of software testing practices for XR applications, including test activities, concerns, and techniques.

RQ2.1: ***What test activities are involved in XR applications?*** This sub-question seeks to identify and categorise the test activities relevant to XR app testing, such as test data generation and test execution. By exploring these activities, we aim to understand the current practices in XR app testing and identify potential areas for improvement.

RQ2.2: ***What are the primary test concerns in XR applications?*** This sub-question focuses on the key concerns in testing XR apps, particularly the *objectives* (e.g., verifying functionality, improving usability) and *targets* (e.g., user interfaces, XR-specific requirements) of testing. Understanding these concerns helps to clarify the goals and challenges in XR app testing.

RQ2.3: ***What test techniques are employed in XR applications?*** This sub-question investigates the specific testing techniques used on XR apps, such as random testing, mutation testing, and model-based testing. By analysing these techniques, we aim to investigate the common testing approaches for XR systems.

RQ3: ***To what extent are XR testing approaches validated?*** This RQ explores how the testing methodologies for XR apps are validated. We assess the metrics and environments (e.g., simulation or real devices) used to evaluate their effectiveness.

4.1.2 Search string

As the research questions aim to investigate the current research status of XR software testing, it is possible that some studies do not directly focus on testing techniques but instead analyse other aspects related to testing. For example, some studies may investigate the characteristics or challenges of XR systems, such as identifying issues or limitations in XR apps, which can indirectly inform testing practices. Specifically, we tackle this by also including studies that explore the nature of *bugs* (or *faults*, etc.) in XR apps, aiming to collect studies analysing them or proposing techniques to detect them. Since XR software testing is still in its early stages, identifying and understanding such issues may still be underexplored.

To ensure that the search process identifies primary studies addressing the RQs, we followed the guidelines by Kitchenham and Charters (2007) to break down the research questions into individual facets using the PICOC model (population, intervention, comparison, outcomes, and context), which then serves as the foundation for designing the search query. The PICOC model is defined in Table 2 and based on this model, the search query for the digital libraries is:

$$\textit{Search String} = (\$XR\ \textbf{AND}\ \$XR_{acr})\ \textbf{AND}\ (\$T\ \textbf{OR}\ \$B)$$

Table 2 PICOC criteria applied to this study

Criterion	Description
Population	XR-related software
Intervention	Testing techniques or relevant studies addressing testing aspects
Comparison	Not applicable
Outcome	Insights into methodologies or practices for testing XR applications
Context	Peer-reviewed publications

Table 3 Synonyms in $\$XR$, $\$XR_{acr}$, $\$T$, and $\$B$

	Synonyms	Metadata
$\$XR$	"virtual reality" **OR** "augmented reality" **OR** "mixed reality" **OR** "extended reality"	title, full text
$\$XR_{acr}$	VR **OR** AR **OR** XR **OR** MR	title
$\$T$	test **OR** detect **OR** detection **OR** verify **OR** verification	title
$\$B$	bug **OR** fault **OR** defect **OR** error	title

Here, $\$XR$ denotes the synonyms of *extended reality*; $\$XR_{acr}$ are the acronyms corresponding to these terms; $\$T$ represents the synonyms of *testing*; and $\$B$ are the synonyms of *bugs*. The synonyms used in the search query are detailed in Table 3.

The search string is searched with the studies' titles, and $\$XR$ is additionally searched with full text to ensure $\$XR_{acr}$ in the titles genuinely referred to extended reality. For example, MR also stands for "magnetic resonance" (Charron et al. 2018), which would yield irrelevant results. By structuring the search string this way, we avoid retrieving extraneous findings related to unrelated fields like medicine.

4.1.3 Search evaluation

To evaluate the quality of the search string, we follow the guidelines of Petersen et al. (2015), using a test set of relevant papers, all of which should be found by the search string. We identified eight studies during our initial literature review (Wang 2022; Wang et al. 2023; Rzig et al. 2019; Rafi et al. 2023; Bierbaum et al. 2003; Corrêa Souza et al. 2018; Li et al. 2020; Andrade et al. 2020) to compose the test set. These studies cover different testing aspects (e.g., functionality, usability, empirical studies), ensuring that the search results include studies with diverse focuses. We refine the search string iteratively until the search results contain all the studies from the test set.

4.1.4 Digital library

To cover as many relevant studies as possible, we conduct our search using OpenAlex[13], a bibliographic database that indexes scientific papers from major digital libraries, including IEEE Xplore Digital Library[14], ACM Digital Library[15], and Scopus[16] (Priem et al. 2022). OpenAlex's filter features allow us to restrict the search to studies in the fields of **Computer Science** and **Engineering**, reducing irrelevant search results Zein et al. 2016; Tramontana et al. 2019. This restriction excludes papers

[13] https://openalex.org/

[14] https://ieeexplore.ieee.org/

[15] https://dl.acm.org/

[16] https://www.scopus.com/

focused on the applications of XR in other disciplines, such as **Medicine** and **Social Sciences**.

4.1.5 Selection criteria

After executing the search string in the digital library, a list of potentially relevant studies is retrieved. To ensure that we only include studies aligned with the mapping study's objectives and capable of answering the research questions, we developed a set of selection criteria (Petersen et al. 2008). As suggested by Petersen et al. (2015), we piloted the selection criteria (using a sample of 100 studies from the search results) and refined them until consensus was reached among the three authors of this paper that the criteria effectively included relevant studies and excluded irrelevant ones. As a result of this process, we applied the following inclusion criteria (ICs) and exclusion criteria (ECs):

$IC1$: Studies must involve software testing techniques, challenges or limitations for extended reality software applications.
$IC2$: Studies published between January 2000 to July 2024.
$IC3$: Studies written in English, published in peer-reviewed journals or conference proceedings, and available in full text.
$IC4$: Studies must be primary studies rather than secondary studies such as systematic literature reviews.
$EC1$: The focus of the studies is not testing but other software development aspects, such as analysis, design or implementation.
$EC2$: Studies do not focus on software-related aspects, such as requirements or integrity, but instead emphasise other areas like user perceptions or hardware configurations.
$EC3$: Studies are duplicated in the search results, including extended versions of existing results.
$EC4$: Studies published in the form of abstract or panel discussion.

4.2 Conducting the mapping

To streamline the methodological information, this subsection provides the fundamental and essential information for conducting this mapping study, including search and selection strategy, classification scheme and data extraction, with more details that can be found at the online repository at https://sites.google.com/view/xr-testing.

4.2.1 Search and selection strategy

The search strategy consists of automated search and manual snowballing. The steps of the search and selection process are illustrated in Fig. 5 and detailed below:

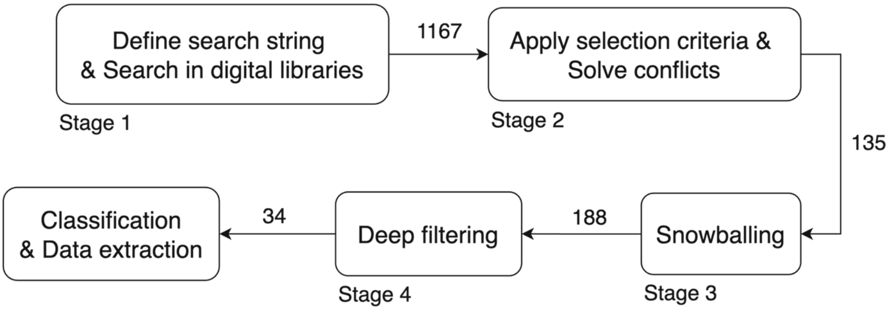


Fig. 5 Literature search and selection process. The numbers on the arrows indicate the number of studies provided to the next stage

- ***Stage 1:*** The search process begins with formulating an initial search string (§4.1.2) to retrieve potential relevant studies from the digital library (§4.1.4).
- ***Stage 2:*** Selection criteria from §4.1.5[17] are applied by reviewing titles and abstracts, with three authors independently evaluating each study and resolving conflicts through consensus meetings to ensure rigour.
- ***Stage 3:*** Backward snowballing is conducted on retained studies following guidelines by Wohlin (2014), with exhaustive iterations performed to ensure comprehensive coverage.
- ***Stage 4:*** Full-text reviews are conducted on remaining studies, focusing specifically on exclusion criteria *E1* and *E2* (§4.1.5) to ensure only truly relevant studies proceed to the classification phase.

After completing this process, the remaining studies form the *primary studies* for this mapping study. These studies proceed to subsequent phases, including classification and data extraction. The full list can be found in Appendix A.

4.2.2 Classification scheme

The classification scheme organises the primary studies into broad categories to provide a structured overview of the field (Kitchenham and Charters 2007). Following the guidelines by Petersen et al. (2015), we applied topic-independent classification, including publication venue and research type, and topic-specific classification.

For topic-specific classification, we utilised the systematic keywording of abstracts method outlined by Petersen et al. (2008), extracting keywords from abstracts (consulting introductions and conclusions when needed) to consolidate them into broader categories. To ensure the reliability of keywording, one author classified all studies, while two others independently classified half each, with disagreements resolved through consensus meetings.

[17] Excluding IC2 and IC3 which are applied automatically using OpenAlex's filtering feature. The full filters applied by OpenAlex can be found at https://bit.ly/3zTxwCS.

4.2.3 Data extraction

To address the RQs outlined in §4.1.1, we systematically extracted data from each primary study, following guidelines by Kitchenham and Charters (2007). We composed a data extraction form (Table 4), including general publication information and research-question-specific data. Additionally, for reproducibility and practical application, we identified testing-relevant datasets and tools referenced in the primary studies.

For clarity and consistency, the data extraction form was pilot-tested with our initial study set (§4.1.3). To mitigate bias, one author extracted data from all primary studies,

Table 4 Data Extraction Form

Data Item	Description (and possible values)	RQ
Title	Title of the study	–
Authors	Names of the study's authors	–
Year	Publication year of the study	*RQ1*
Venue	Name of the publication venue	–
Venue Type	Type of the venue (e.g., conference, workshop, journal)	*RQ1*
Topic	Primary focus area of the study (e.g., usability testing, automated testing)	*RQ1*
Research Type	Research type of the study (e.g., solution proposal, validation research)	*RQ1*
Technology	Immersive technology specified in the study (e.g., XR, VR, AR)	*RQ1*
Test Activity	Specific Test activity involved (e.g., test data generation, test tool development, test execution)	*RQ2.1*
Test Objective	Primary objective of the testing approach (e.g., functionality, usability, security)	*RQ2.2*
Test Target	Focus of the testing approaches (e.g., general, GUI, XR-specific requirements)	*RQ2.2*
Test Level	Scope of the testing activities (i.e., unit testing, integration testing, system testing)	*RQ2.3*
Test Type	Type of testing performed (e.g., black box, white box)	*RQ2.3*
Test Technique	Core methodologies used for testing (e.g., search-based testing, mutation testing)	*RQ2.3*
Evaluation Environment	Environment for evaluating the testing approaches (e.g., Unity Editor, HMD, mobile device)	*RQ3*
Metrics	Metrics used to evaluate testing techniques (e.g., coverage, mutation score)	*RQ3*
$Dataset_{train}$	Details of datasets used for training machine learning-based approaches, including content types (e.g., video, image) and dataset size	*discuss.*
$Dataset_{eval}$	Details of datasets for evaluating testing techniques, including content types and size	*discuss.*
Tool	Details of software tools proposed or used by the study	*discuss.*

with the other authors reviewing the results, and any disagreements were resolved through consensus meetings.

After the classification and data extraction processes, we analysed and formulated the retrieved data to address the RQs. The analysis results are presented in §5.

4.3 Reporting the mapping

The report contains two parts. The first part comprises this paper, which outlines the study's methodology and findings, and the second part is the data extraction form, which details the raw data collected and the basis for the study's conclusions. The complete results of the classification and data extraction are publicly accessible at: https://sites.google.com/view/xr-testing.

4.4 Threats to validity

This section addresses potential threats to the completeness of the literature search and selection process. This threat is influenced by the search string choice, the bibliographic database limitations, and the robustness of the literature selection process.

To mitigate the risk of excluding relevant studies during the selection process, three authors independently screened all search results, following the most inclusive approach. Conflicts were resolved through consensus discussions. While the inter-rater agreement was not formally measured, our approach prioritised achieving absolute consensus to maximise the inclusion of relevant studies.

We acknowledge potential limitations in our search strategy–solely using OpenAlex as the digital library–may affect the thoroughness of our study. While OpenAlex indexes publications from major digital libraries such as IEEE, ACM, and Scopus, we recognise that it does not index some relevant studies (e.g., those that may exist exclusively in specialised venues or databases). Additionally, very recently published studies might not have been indexed in OpenAlex at the time of our search, creating a temporal bias against the latest research.

To mitigate these risks, we adopted an exhaustive iterative snowballing strategy. We systematically identified relevant studies from the reference lists of each included study and repeated this process until no new relevant studies were discovered. Our snowballing process involved three iterations, which significantly reduced the likelihood of missing important contributions to the field.

5 Results

In this section, we present the results of this mapping study, including information about the search and selection results of the primary studies, and answer the research questions based on the information from the primary studies.

The initial search returned 1167 studies from OpenAlex. The selection process, as outlined in §4.2.1, reduced this to 135 studies after applying the selection criteria. In parallel with deep filtering, backward snowballing identified 53 additional studies, resulting in a final set of 34 primary studies for this mapping study (see Appendix A for the complete list of primary studies).

Figure 6 shows the publication trend of the primary studies from January 2000 to July 2024. The data reveal that the XR software testing research field was relatively inactive before 2017, with only two studies published. However, starting in 2017, the number of studies began to increase gradually, reaching a peak in 2023, with ten studies published that year. As the literature search for this study was conducted in July 2024, the number of studies published in 2024 was not completely recorded.

5.1 RQ1: research status

To address RQ1, which explores the current status of research in XR software testing, we present the classification results of the primary studies. Additionally, we analyse the immersive technologies (e.g., AR, VR) featured as testing subjects in these studies, providing insights into the technologies most frequently explored in this domain.

The studies are classified based on the following criteria: (1) venue types (e.g., conferences, journals), (2) study topics (e.g., automated testing, usability testing), and (3) research types (e.g., solution proposal, evaluation research).

5.1.1 Venues

The distribution of primary studies by venue type is presented in Fig. 7. Conferences emerge as the dominant venue type, accounting for about 38% of the primary studies.

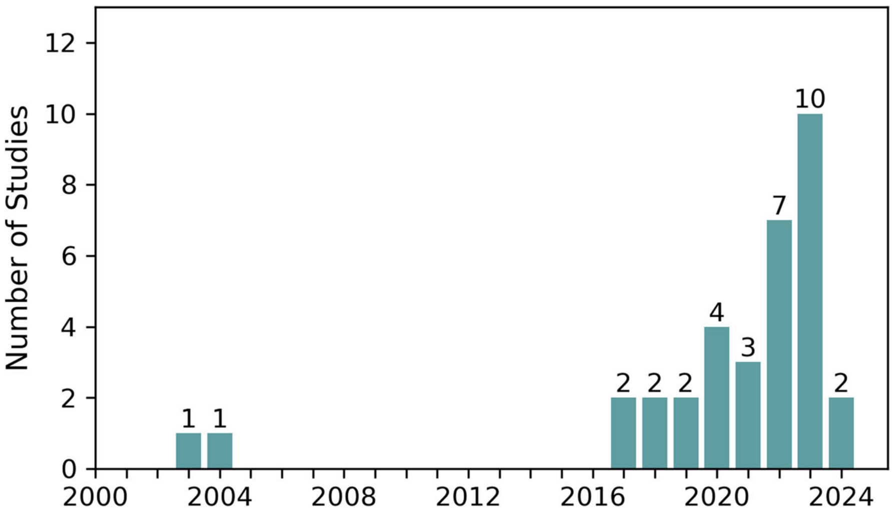


Fig. 6 Publication years of primary studies

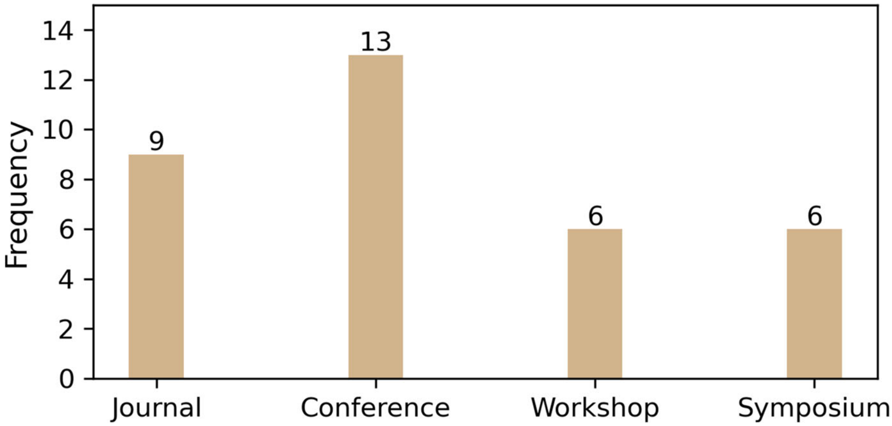


Fig. 7 Distribution of publication venue types for the primary studies

Journals are the second most common venue, publishing approximately 26% of the studies. Workshops and symposiums each represent about 18% of the total primary studies[18]. The complete list of all publication venues for the primary studies is available at: https://sites.google.com/view/xr-testing.

5.1.2 Topics

As explained in §4.2.2, we carefully categorise the primary studies into eight topics. Table 5 presents the resulting classification. Overall, *automated testing* emerges as the most prominent topic, with seven studies (21% of the total). *Usability testing* and *security testing* each account for six studies (18%). *XR-specific testing* and *scene testing*, each contribute four studies (12%), while *test automation* and *open-source projects* each comprise three studies (9%). Finally, *stakeholder survey* represents the least explored topic, with only one study (3%).

Below we provide details for each topic, including: (1) *description:* a general overview of the topic, (2) *examples:* summaries of representative studies within the topic, and (3) *implications:* insights into the potential consequences, proposed solutions, or general guidelines for addressing the challenges associated with the topic.

XR-specific Testing

- **Description:** Testing techniques for XR-specific requirements like collision, occlusion, registration, and tracking (Doerner et al. 2022).
- **Examples:** Collision and occlusion are critical real-time requirements in XR apps, where the former refers to the interaction between objects when they come into contact and the latter occurs when objects block each other from view Breen

[18] The distinction between symposiums and conferences is not always clear. For classification purposes, venues with “conference" in their tile are categorised as conferences and those with “symposium" are classified as symposiums

Table 5 Research topics in the primary studies

Topic	Primary studies	Number
XR-specific testing	PS16, PS17, PS20, PS22	4
Scene testing	PS4, PS12, PS32, PS33	4
Security testing	PS2, PS13, PS14, PS15, PS18, PS30	6
Usability testing	PS1, PS7, PS10, PS11, PS19, PS21	6
Automated testing	PS3, PS5, PS8, PS23, PS24, PS25, PS27	7
Test automation	PS9, PS26, PS34	3
Open-source projects	PS6, PS28, PS31	3
Stakeholder survey	PS29	1

et al. 2000; Doerner et al. 2022. Testing these aspects ensures realistic interactions between virtual and real objects. PS17 Andrade et al. 2023 proposes an approach that automatically generates test data to detect incorrect collision and occlusion in VR apps.
- **Implications:** Testing XR-specific requirements demands a deep understanding of their impact on software behaviour, and tailored testing techniques for these unique challenges.

Scene Testing

- **Description:** Validates XR functionality through exploration of XR scenes and interaction with virtual objects.
- **Examples:** PS32 (Wang et al. 2023) and PS33 (Wang 2022) introduced VR scene testing techniques, focusing on exploring environments, triggering interactable objects, and optimising interaction routes.
- **Implications:** Scene testing extends principles of 2D GUI testing into more complex 3D environments with 6DOF interactions.

Usability Testing

- **Description:** Identifies usability issues (e.g., side effects) in XR software.
- **Examples:** PS1 (Jung et al. 2017), PS19 (Li et al. 2024), and PS21 (Kim et al. 2017) proposed approaches for detecting *cybersickness*, a prevalent side effect in VR systems using visual content (e.g., screenshots) analysis.
- **Implications:** While most existing research employs user-centric methods, such as user studies (§3.1), understanding the root causes of these usability issues would enable the development of systematic software-centric approaches.

Security Testing

- **Description:** Identifies and mitigates *security* and *privacy* issues in XR software. Security testing detects system intrusion and addresses vulnerability, while privacy testing protects users' sensitive information.

- **Examples:** PS18 (Lehman et al. 2022) introduces a framework to address privacy issues in mobile AR apps, while PS15 (Valluripally et al. 2023) targets attacks that disrupt VR user experiences.
- **Implications:** Security and privacy testing should account for XR-specific features, such as interaction with physical environments and differentiation between real and virtual objects (Casey et al. 2021).

Automated Testing

- **Description:** Automates both test *generation* and *execution* for functional testing, without specifically targeting XR-specific or non-functional requirements.
- **Examples:** PS23 (Rafi et al. 2023) and PS27 (Yang et al. 2024) tackle the oracle problem of object misplacement in AR apps using neural networks to detect errors in screenshots depicting object misplacement scenarios.
- **Implications:** Effective test automation often requires a thorough understanding of system requirements or test oracles, which is essential for systematically generating reliable test data.

Test Automation

- **Description:** Automating test execution but not test generation.
- **Examples:** PS34 (Figueira and Gil 2022) presents a unit testing framework for Unity-based VR/AR apps using manually created test scripts.
- **Implications:** Test automation often involves script-based testing frameworks for automated execution driven by predefined tests.

Open-source Projects

- **Description:** Empirical studies that analyse open-source XR projects to gain insights into current testing practices and challenges.
- **Examples:** PS6 (Li et al. 2020) examines bugs in open-source WebXR projects to explore bug symptoms and root causes, while PS31 (Rzig et al. 2019) investigates open-source VR projects and reveals their insufficiency in testing.
- **Implications:** Studying open-source projects provides valuable empirical data and practical recommendations for the emerging field of XR software testing.

Stakeholder Survey

- **Description:** Surveys and interviews with real-world stakeholders like XR users and developers, gathering perspectives on testing practices and challenges.
- **Examples:** PS29 (Andrade et al. 2020) surveys XR stakeholders to understand software testing practices, highlighting key concerns and common faults, such as interaction issues and crashes.
- **Implications:** Stakeholder surveys provide real-world insights, complementing open-source project analysis and guiding testing improvements.

5.1.3 Research types

For research types, we adopt the classification categories proposed by Wieringa et al. (2006): *solution proposal*, *validation research*, *evaluation research*, *philosophical paper*, *opinion paper*, and *experience paper*. These categories have been carefully reviewed and adapted to align with the scope of XR software testing, ensuring relevance to the primary studies. Notably, a study can span multiple categories, such as studies that propose solutions and include initial validation.

Table 6 summarises the results. *Solution proposal and validation research* emerges as the most prevalent research type, encompassing 15 studies (44% of the total). *Solution proposal* accounts for 6 studies (18%), *evaluation research* includes 5 studies (15%), and *solution proposal and evaluation research* covers 4 (12%). On the other hand, *validation research* and *philosophical papers* each account for 2 studies (6%). Notably, no *opinion paper* and *experience paper* were identified in the primary studies.

Similar to the approach in §5.1.2, we present the *description* and *examples* of each research type based on the primary studies.

Validation research

- **Description:** Providing initial validations of solutions or problems, typically involving limited experiments in controlled, simplified settings, such as toy applications (e.g., research prototypes or low-popularity open-source apps) and datasets of minimal complexity.
- **Examples:** PS12 (Gunawan et al. 2023) introduces a black-box testing approach for a VR musical instrument game, using equivalence partition to design the test cases. Validation was limited to manual assessment of test results without systematic methodologies.

Evaluation Research

- **Description:** Conducts rigorous testing in real-world settings, addressing meaningful research questions. These studies engage real users or practitioners or evaluate with practical applications, such as industrial software or widely used open-source projects, and using datasets derived from real-world scenarios.
- **Examples:** PS31 (Rzig et al. 2019) conducted an empirical study on VR automated testing in open-source VR projects, revealing gaps in current practices.

Table 6 Research types in the primary studies

Research type	Number of studies
Validation research	2
Evaluation research	5
Solution proposal	6
Solution proposal *and* validation research	15
Solution proposal *and* evaluation research	4
Philosophical papers	2

Solution Proposal

- **Description:** Proposes innovative approaches to XR testing challenges, focusing on theoretical benefits with minimal empirical evidence. These studies typically use basic examples and lack experimental validation with real-world applications.
- **Examples:** PS3 (Prasetya et al. 2021) presents an autonomous agent-based testing framework for XR systems. The study details the architecture and potential applications but without experimental assessment.

Solution Proposal and Validation Research

- **Description:** Combines proposing novel solutions with preliminary validation, typically in simplified experimental settings.
- **Examples**: PS23 (Rafi et al. 2023) presents a technique for detecting object misplacement issues in AR apps. It is validated using Unity-provided examples rather than real-world apps.

Solution Proposal and Evaluation Research

- **Description:** Proposes novel solutions and rigorously evaluates them in real-world contexts.
- **Examples**: PS19 (Li et al. 2024) introduces a technique to detect stereoscopic visual inconsistencies in VR apps, validated using screenshots from real-world VR apps available on the Steam store.

Philosophical Papers

- **Description:** Focuses on theoretical perspectives or conceptual frameworks rather than implementing technical solutions. These studies aim to propose new ways of thinking about challenges, without presenting concrete implementations.
- **Examples:** PS13 (Kilger et al. 2021) outlines general guidelines for detecting and preventing cybersecurity attacks in MR environments, identifying threats and countermeasures but offering no implementations.

The majority of primary studies, 25 out of 34 (74%), propose novel solutions for XR software testing problems. Among these, 15 (60%) include only basic validation, highlighting the emerging nature of the field, with limited research evaluated in real-world scenarios.

The first empirical evaluation study was published in 2019 (PS26). More recently, 2023 saw the introduction of two novel testing solutions evaluated in real-world contexts (PS16 and PS32). This trend suggests increasing potential for applying XR testing techniques in practical, real-world environments in the near future.

5.1.4 Immersive technology

We analyse the primary studies by examining the specific immersive technologies targeted for testing in these studies. Figure 8 summarises the technologies examined

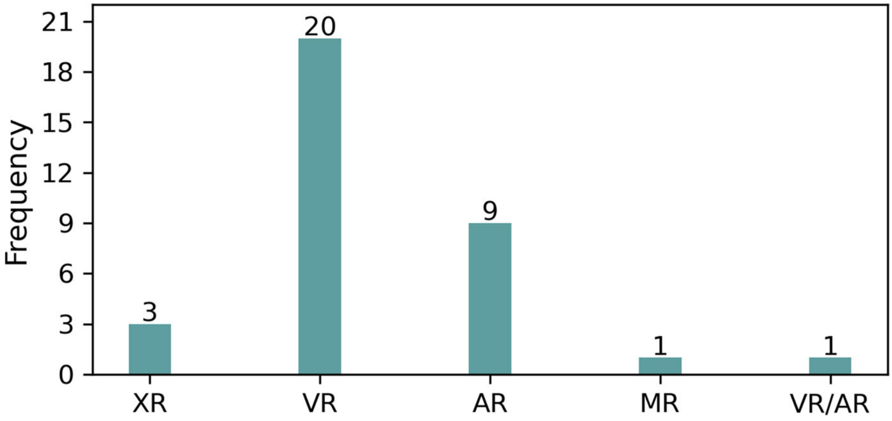


Fig. 8 Immersive technologies of primary studies

in the primary studies. About 59% of the studies focus on testing VR apps, while approximately 26% target AR apps. Only a small number of studies explore broader or integrated scopes: three focus on XR systems[19], one investigates MR testing, and one addresses both VR and AR testing.

The findings indicate that while VR and AR testing have received significant research attention, studies addressing broader scopes are still in the early stages of development within the research landscape.

> **The answer to RQ1, i.e., the current status of XR application testing research, is as follows:**
> **Publication trends:** Research on XR software testing has grown steadily, increasing from 2 publications in 2017 to 10 in 2023.
> **Publication venues:** Conferences and journals are the main publication venues, representing 38% and 26% of the studies, respectively.
> **Research topics:** Automated testing is the most prevalent research topic, accounting for 21% of studies, followed by usability testing and security testing, each contributing 18%.
> **Research types:** 74% of studies propose novel XR testing solutions, with 60% relying on preliminary validations in controlled settings.
> **Immersive technologies:** VR dominates with 59% of studies and AR represents 26%. XR, MR, and cross-technology research contributes 15%.
> In summary, XR software testing is an emerging field, steadily gaining momentum.

[19] We acknowledge XR is an umbrella term that includes VR, AR, and MR; this categorisation is based on each study' specific context and terminology.

5.2 RQ2: testing facets

We classify the primary studies based on three key test facets: *test activities* (e.g., test generation, test execution), *test concerns* (including objectives like functionality and security, and targets such as user interaction and collision), and *test techniques* (e.g., random testing, model-based testing). To ensure the meaningfulness of the extracted information, we exclude the studies that do not directly yield testing facets, which are five studies identified as empirical studies (PS2, PS6, PS28, PS29, PS31) and two classified as philosophical papers (PS13 and PS24). The remaining 27 studies are analysed to address this research question.

5.2.1 RQ2.1: test activities

Figure 9 visualises the distribution of test activities in the primary studies using a word cloud. The size of each keyword in the word cloud corresponds to its frequency, with larger keywords appearing more frequently in the test activities.

For instance, “test" is the most prominent keyword, reflecting its centrality across various activities. Among these, “generation" and “automation" are the next most prominent keywords, indicating that activities such as test generation and test automation are the most frequently addressed activities in the studies.

11 studies include the keywords “test” and “automation”, and all are associated with the test activity *test automation*. On the other hand, eight studies include the keywords “test" and “generation", among them, three involve the activity *test generation*, and five involve *test input generation*. The distinctions between the three activities are: (1) *test automation* only automates the execution of tests and does not include generating test data or oracles; (2) *test input generation* involves creating test input data, which can be done either manually or automatically, but it does not include test oracle generation; (3) *test generation* automates the creation of both test inputs and test oracles. Among

Fig. 9 Word cloud based on the test activities of the primary studies

these activities, *test generation* is the least explored, appearing in only three studies, indicating its higher technical challenges compared to the other test activities.

Other recurring activities include *oracle prediction*, *test execution*, and *attack detection*. These findings highlight a clear focus on minimising manual effort and enabling scalable testing for XR software.

5.2.2 RQ2.2 test concerns

Test concerns cover both *test objectives* (e.g., functionality, usability, security) and *test targets* (e.g., cybersickness, collision) of testing.

Test Objective We categorise the primary studies into six groups based on their test objectives: (1) *functionality:* testing whether the functional specifications are correctly implemented, (2) *usability:* assessing whether SUT negatively impact user experience, (3) *security:* ensuring the SUT is protected from external attacks, (4) *privacy:* verifying that user's personal data is safeguarded against local threats, (5) *performance:* checking whether the SUT meets specific performance requirements (e.g., response time), and (6) *load:* evaluating the SUT's behaviour, reliability, or stability under stress.

Table 7 provides an overview of the test objectives. Among these, *functionality* is the most common objective in the primary studies. It is worth noting that individual studies can cover multiple test objectives. For instance, PS7 (Lehman et al. 2023) addresses *functionality*, *usability*, and *performance*. It proposes a testing framework for system testing of mobile AR apps. The framework provides features such as collecting usability information, including the quality of the user experience in AR scenes; monitoring performance metrics, such as frames per second (FPS) traces, to identify performance dips; detecting functional edge cases through long-term monitoring.

Test Target To illustrate the relationship between test objectives and their associated test targets, we present a bubble chart highlighting how specific test targets align with certain test objectives. Importantly, a single study can address multiple test targets under a single test objective. For example, PS17 focuses on the test objective *functionality* and includes the test targets *collision* and *occlusion*.

Table 7 Test objectives in the primary studies

Test objective	Primary Studies	Number
Functionality	PS3, PS4, PS5, PS7, PS8, PS9, PS12, PS16, PS17, PS22, PS23, PS25, PS26, PS27, PS32, PS33, PS34	17
Usability	PS1, PS7, PS8, PS10, PS11, PS19, PS20, PS21	8
Security	PS14, PS15, PS30	3
Privacy	PS15, PS18	2
Performance	PS7, PS8	2
Load	PS8	1

Figure 10 shows the bubble chart. Among the test objectives, *functionality* is the most comprehensive, covering nine distinct test targets. The most studied target under this objective is *user interaction*, which focuses on testing interaction features in XR systems and has been examined in five studies. Other notable targets include *general*, which are the general guidelines for testing XR systems, and *scene exploration*, where testing focuses on exploring the XR scenes, each represented by three studies.

The second most prevalent test objective, *usability*, encompasses five test targets. Among these, *cybersickness* is the most studied, appearing in three studies, while *user interaction* is addressed in two studies. These findings align with the critical importance of user experience and the operational and interactional aspects of XR systems, emphasising the primary focus on XR software testing efforts.

The other test objectives, i.e., *security*, *privacy*, *performance*, and *load*, cover fewer test targets. Both *security* and *privacy* are linked to two test targets each, while *performance* and *load* are associated with only one test target each.

This analysis underscores the diversity of test targets within each objective and highlights relatively well-explored areas versus those requiring further investigation.

5.2.3 RQ2.3 test techniques

We address this sub-question by analysing the test level (e.g., system testing, unit testing), test type (i.e., black-box or white-box testing) and the specific techniques employed (e.g., random testing, search-based testing).

Test Level A single study can address multiple test levels. For instance, PS9 introduces preliminary solutions for both unit testing and system-level interaction in VR systems.

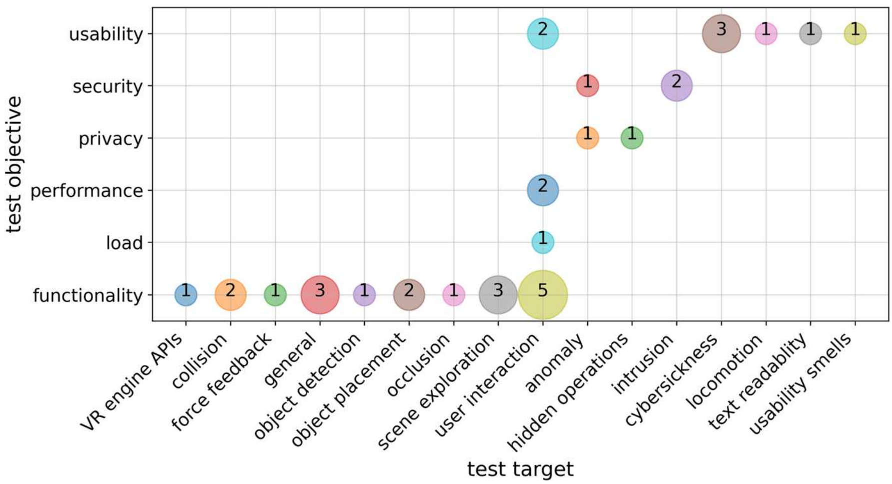


Fig. 10 Bubble chart of test objectives against test targets

Table 8 provides an overview of the test levels explored in the primary studies. *System testing* is the most prominent test level, featured in 22 studies. In contrast, *unit testing* and *non-functional testing* each appear in only three studies, while *integration testing* is represented by a single study.

Test levels do not apply to PS16. The study proposes a technique for recommending potential test types (i.e., animation, colliding, and general) for VR systems but does not explicitly perform any testing actions. Therefore, PS16 is excluded from the data extraction for both *test level* and *test type*.

Test Type We define *white-box testing* as testing that requires access to the source code of the SUT, such as instrumentation or static analysis. In contrast, *black-box testing* does not involve internal information about the SUT. Instead, it relies on analysing the system's input and output behaviour, such as evaluating screenshots or video recordings of specific actions in the XR systems. Notably, a single study may include both black-box and white-box testing methods.

In our analysis, we excluded three studies categorised under the *non-functional testing* level (discussed in the previous paragraph) from the extraction of test types. This is because black-box and white-box testing typically focus on verifying the *functionality* of the SUT.

Table 9 presents a summary of the findings. The distribution studies utilising *black-box testing* and *white-box testing* are relatively balanced, with 13 and 10 studies, respectively. In addition to the four excluded studies, two studies lack sufficient evidence to determine their test type and are therefore categorised as *unclear*. newline

Test Techniques Table 10 summarises the test techniques utilised in the primary studies. A few studies that do not employ specific testing techniques but instead provide general testing guidelines are excluded from the table. Additionally, a single study may apply multiple test techniques. In total, 14 distinct techniques are identified, with two being adopted by multiple studies: *machine learning-based testing*, used in nine studies, and *model-based testing*, applied in four studies. These findings align with

Table 8 Test levels in the primary studies

Test Level	Primary Studies	Number
System Testing	PS1, PS3, PS4, PS5, PS7, PS8, PS9, PS10, PS11, PS12, PS17, PS18, PS19, PS20, PS21, PS22, PS23, PS25, PS26, PS27, PS32, PS33	22
Integration Testing	PS26	1
Unit Testing	PS9, PS26, PS34	3
Non-functional Testing	PS14, PS15, PS30	3
Not Applicable	PS16	1

Table 9 Test types in the primary studies

Test Type	Primary Studies	Number
Black-box Testing	PS1, PS11, PS12, PS17, PS18, PS19, PS20, PS21, PS22, PS23, PS25, PS26	13
White-box Testing	PS3, PS4, PS7, PS9, PS10, PS26, PS27, PS32, PS33, PS34	10
Unclear	PS5, PS8	2
Not Applicable	PS14, PS15, PS16, PS30	4

the characteristics of XR systems. Machine learning techniques are particularly well-suited for handling the rich graphical interfaces of XR systems, such as identifying faults using app screenshots. Meanwhile, model-based testing simplifies the inherent complexity of XR systems by abstracting them into models, facilitating systematic testing.

To further investigate the most predominant test technique, machine learning-based testing, we examine the dataset used in these studies to train the ML models for testing in §6.2.

Although the remaining test techniques are each represented by only one study, this diversity highlights the successful exploration and adoption of various innovative approaches in the emerging field of XR software testing. Future research could explore the underrepresented areas like integration testing and expand the application of emerging test techniques.

Table 10 Test techniques in the primary studies

Test Technique	Primary Studies	Number
Static Analysis	PS4	1
Dynamic Analysis	PS4	1
Statistical Analysis	PS15	1
Image Processing	PS1	1
Machine Learning	PS11, PS14, PS15, PS17, PS19, PS21, PS23, PS27, PS30	9
Model-based Testing	PS3, PS4, PS5, PS10	4
Search-based Testing	PS32	1
Mutation Testing	PS25	1
Metamorphic Testing	PS17	1
Random Search	PS33	1
Greedy Search	PS33	1
Record and Playback	PS25	1
Runtime Monitoring	PS18	1
Equivalence Partition	PS12	1

The answer to RQ2, i.e., the test facets involved in XR applications, is as follows:
Test activities: The most common test activities involve test automation (11 studies) and test generation (8 studies), reflecting a focus on reducing manual effort in testing XR software.
Test concerns: The primary test objectives are functionality (17 studies) and usability (8 studies). The most frequent test targets for functionality are user interaction (5 studies), while for usability, the key focus is cybersickness (3 studies). These align with critical user experience and interactional aspects of XR systems.
Test techniques: System testing is the dominant test level (22 studies), with black-box and white-box testing nearly balanced. Machine learning is the most prevalent technique (9 studies), followed by model-based testing (4 studies).

5.3 RQ3: evaluation

We address RQ3 by presenting the evaluation metrics reported in the studies and the evaluation environment used to assess the testing techniques.

Several studies lack concrete evaluation, as they do not fall under the research types of *evaluation research* or *validation research* (§5.1.3). Specifically, six solution proposals, five empirical studies and two philosophical papers are excluded from the extraction of evaluation metrics and environments.

5.3.1 Metrics

Table 11 summarises the evaluation metrics used in the primary studies. As discussed in §5.2.3, ML-based testing is the most prevalent technique, represented by nine studies. Among these, seven studies used *standard ML metrics*, such as *precision*, *recall*, and *F1-score*, to evaluate ML performance. One study relies on *classification results*, which provide general information on the number of correctly classified cases.

The studies identify four kinds of coverage metrics: (1) *method coverage:* measures the percentage of methods exercised by tests out of the total number of methods; (2) *model coverage:* calculates the proportion of states covered by tests in a model as used in model-based testing, such as a finite state machine; (3) *requirement coverage:* computes the percentage of nodes covered in a requirement flow graph, which is derived from an XR app's scene graph; (4) *object coverage*, assesses the percentage of interactable objects triggered by tests. Coverage metrics are utilised in four studies, with one covering both method and object coverage. Similarly, *manual validation*, where results are manually verified, is also used in four studies.

Table 11 Evaluation metrics in the primary studies

Metrics	Primary Studies	Number
Standard ML Metrics	PS11, PS14, PS17, PS18, PS19, PS23, PS27, PS30	8
Classification Results	PS15	1
Manual Validation	PS10, PS12, PS16, PS21	4
Method Coverage	PS32	1
Model Coverage	PS4	1
Object Coverage	PS32, PS33	2
Requirement Coverage	PS5	1
Mutation Score	PS5, PS25	2
SSQ Score	PS21	1
Detected Bugs	PS32	1
Object Detection Success	PS22	1
Suspiciousness Score	PS15	1

However, the reliance on manual validation suggests the need for more systematic and automated evaluation frameworks, especially as XR systems grow in complexity.

5.3.2 Evaluation environment

To explore what types of environments are involved in the evaluation, such as simulation, mobile devices, and HMDs, we investigate the evaluation environments within the studies. Table 12 provides the results of evaluation environments. The most common evaluation environments are *HMD* (head-mounted display) and *Unity Editor*, each represented in six studies. For clarity, we define the environment *Unity Editor* as the simulation performed within the Unity's *Scene* or *Game* view[20]. *Mobile device* is the second most common environment, used in four studies, while *haptic device* and *cloud* are the least common, each appearing in one study.

Besides the studies excluded from the extraction of evaluation metrics and environments due to their research types or empirical nature, five studies lack enough details to determine the evaluation environment and are therefore labelled as *unclear*.

The focus on HMDs, Unity Editor, and mobile devices underscores their critical role in real-word testing, while the limited diversity in environments such as cloud-based or haptic devices, suggests opportunities for further exploration and innovation.

Furthermore, we notice some studies utilise existing datasets for evaluation. The details of these datasets (content types, sizes and availability) are discussed in §6.2.

[20] https://docs.unity3d.com/Manual/UsingTheEditor.html

Table 12 Evaluation environments in the primary studies

Evaluation Environment	Primary Studies	Number
HMD	PS10, PS12, PS14, PS21, PS26, PS30	6
Unity Editor	PS16, PS17, PS23, PS26, PS32, PS33	6
Mobile Device	PS18, PS22, PS26, PS27	4
Haptic Device	PS25	1
Cloud	PS15	1
Unclear	PS4, PS5, PS7, PS11, PS34	5

The answer to RQ3, i.e., the extent of the testing approaches validated, is as follows:
Out of 27 studies involving valid test activities, six studies do not provide any evidence on evaluation, leaving 78% of the studies validated through some form of evaluation.
Metric: The most common evaluation metrics are standard machine learning metrics, for evaluating machine learning-based techniques. Additionally, manual validation and different types of coverage metrics are equally prevalent.
Environment: The most frequently used evaluation environment HMD, Unity Editor, and mobile device, reflecting the typical platforms for XR application development and testing.

6 Discussion

In this section, we discuss the findings and implications of this mapping study. Specifically, we address (1) the key insights and lessons learned from our methodology; (2) the datasets and tools utilised or proposed in the primary studies; (3) the implications for practitioners; and (4) the remaining challenges and future research directions identified through our analysis.

6.1 Mapping study methodology

While conducting this mapping study, we carefully considered methodological choices that could influence our findings. Our approach embraces the diverse nature of XR testing research while acknowledging its potential impacts on interpretation.

While differences between research types or publication venues may yield varying depths of evidence, this diversity enhances the value of our mapping study. By capturing the full spectrum of XR testing research, we provide a more accurate representation of the field's current state.

Following the guidelines by Petersen et al. (2015), we deliberately chose an inclusive approach without applying quality assessments during selection. We acknowledge this introduces certain limitations as analysing heterogeneous studies collectively may obscure category-specific characteristics. Despite potential influences on the interpretation of trends, we believe the benefits of comprehensive coverage outweigh these limitations for mapping the emerging research area of XR testing.

6.2 Datasets and Tools

To facilitate future research and practices in XR testing, we present an in-depth investigation of the datasets and tools identified in our primary studies.

This subsection examines (1) datasets used for training ML models in ML-based testing techniques, (2) datasets for evaluating testing techniques, (3) industrial tools employed or referenced in the studies, and (4) research tools used or proposed within the studies. The availability of the datasets and tools is determined as of the submission date of this mapping study (December 2024). The resources are organised and can be accessed at https://sites.google.com/view/xr-testing.

6.2.1 Datasets for training

As discussed in §5.2.3, nine primary studies (PS11, PS14, PS15, PS17, PS19, PS21, PS23, PS27, PS30) utilised ML-based techniques for testing XR apps. To better understand their capabilities and provide valuable resources for future research and practice, we analyse the datasets used for training the ML-based techniques. Among the nine studies, six provide detailed dataset information. We examine their content type, training set size (excluding test sets), data source, and availability, and summarise our findings in Table 13.

Table 13 Training datasets used for machine learning-based testing approaches

Study	Content	Size	Source	Avail.
PS11	Image	600	experiments	F
PS19	Image	20,000	Steam	F
PS21	Video	61	UCSD Ped1 & Ped2, Avenue, KITTI	T
PS23	Image	720	Unity Mars	T
PS27	Image	$\sim$ 2740	Google Play & GitHub	F
PS30	Traffic & attacks	$\sim$ 848,000	CIC-IDS2017	T

Dataset PS11 consists of 600 images of XR scenes, containing some texts in their background. Each image is labelled whether the text is readable or not by human participants and features various configurations of font styles and background textures. The dataset is not publicly available.

Dataset PS19 is a subset of 20,000 stereoscopic screenshots, randomly sampled from an original training set of 154,566 screenshots, collected from 288 VR apps on Steam[21]. Steam is one of the largest platforms for video games, including VR apps.

Dataset PS21 is based on multiple datasets, comprising a total of 61 video clips, each containing 200 frames, to train a model for measuring exceptional motion in VR video content that contributes to cybersickness. The original datasets are UCSD Ped1 and Ped2 Mahadevan et al. 2010, Avenue datasets Lu et al. 2013, and KITTI benchmark datasets Geiger et al. 2013, all are publicly available.

Dataset PS23 consists of 720 screenshots from a basic AR scene provided by Unity Mars[22], a Unity extension for AR/MR content development. The dataset is labelled via crowdsourcing to identify realistic object placement. It is used to train a model to identify object misplacement issues in AR systems, capturing variations in placement gaps, distance, and viewing angles.

Dataset PS27 includes 3043 screenshots from 21 AR apps sourced from the Google Play Store and GitHub. With 90% (approximately 2740 screenshots) allocated for training a model to detect object misplacement issues in AR systems. The dataset is labelled via crowdsourcing to provide placement information. However, the exact numbers of screenshots in the training and testing subsets are not specified in the paper, and the dataset is currently not publicly accessible.

Dataset PS30 utilised the Intrusion Detection Evaluation Dataset (CIC-IDS2017)[23], containing over 2.8 million network traffic instance, including normal traffic and attacks like DoS and DDoS. Reformatted for binary classification (attack vs benign), it comprises 1,211,327 instances, 70% are used for training. Notable discrepancies in reported sample sizes between subsets, therefore the training set size is (70% of 1,211,327, which is approximately 848,000) recalculated for consistency.

Overall, the prevalence of image-based training datasets highlights the potential of image-based techniques to address a wide range of software testing tasks for XR applications effectively.

6.2.2 Datasets for evaluation

This section focuses on evaluation datasets, potentially encompassing diverse data points or scenarios, offering broader applicability for testing methodologies, empirical studies, and potential reuse in future research. Isolated research prototypes or limited open-source applications are not considered comprehensive datasets.

As discussed in §6.2.1, ML-based techniques often evaluate their performance using test datasets, i.e., subsets derived from the same datasets as their training data. Detailed information about these evaluation sets is omitted to avoid redundancy, as

21 https://store.steampowered.com/

22 https://unity.com/products/unity-mars

23 https://www.unb.ca/cic/datasets/ids-2017.html

Table 14 Industrial tools in the primary studies. OSS indicates if the tool is open-source or not

Name	Platform	Input	Test Type	OSS
UTF	Unity	Test scripts	Unit	T
XRI	Unity	Interaction designs	N/A	T
Airtest	Unity, Cocos[26]	Test scripts	Scene	T
AltUnity Tester	Unity, Unreal	Test scripts	Scene	F
ML-Agents	Unity	Training env.	Scene	T
clumsy	Windows	N/A	Network	T
Wireshark	Windows, Linux, macOS	N/A	Network	T

they may only differ from the training sets in size. Apart from these, most studies utilised research prototypes or basic open-source applications.

Two empirical studies, PS2 and PS6, present independent datasets for evaluation. **Dataset PS2** consists of 390 mobile AR apps from the Google Play Store to conduct an empirical study on user privacy concerns in mobile AR apps. However, this dataset is not available. **Dataset PS6** collects 368 real bugs from open-source WebXR projects, labelled with their bug symptoms and root causes and is publicly accessible.

We want to know that multiple studies (PS16, PS17, PS31, PS32, PS33) utilised a dataset called **Unity List**, which is no longer accessible[27].

6.2.3 Industrial tools

Table 14 highlights industrial tools used or referenced in the primary studies. These tools address various testing needs, including GUI, unit, and network testing, as well as one tool for XR interaction development. For each tool, we outline key details such as supported platforms and engines, input formats, test types, and whether the tool is open-source. This information is intended to guide researchers and practitioners in selecting tools suitable for their testing requirements.

Unity Test Framework (UTF)[28] is an official testing tool provided by Unity for unit testing Unity-based projects. It integrates with NUnit [29], a unit testing library for .NET languages.

XR Interaction Toolkit (XRI)[30] is an official Unity package for creating 3D and UI interactions in VR/AR experiences. While it does not directly facilitate XR app testing, it is useful for prototyping research apps that can serve as experimental platforms for testing methodologies.

27 According to Unity List's X homepage https://x.com/unitylist, it is no longer available.

28 https://docs.unity3d.com/Packages/com.unity.test-framework@1.1/manual/index.html

29 https://nunit.org/

30 https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0

Airtest[31] is a visual-based UI test automation framework commonly used for video game testing. It uses screenshot-based locators in test scripts to simulate user actions, making it suitable for dynamic and visually complex interfaces.

AltUnity Tester[32] is a test automation framework designed for games and 3D apps, supporting UI and functional testing. Test scripts interact with Unity elements using identifiers such as object names and tags, simulating user actions.

ML-Agents[33] is an open-source toolkit by Unity for training intelligent agents in Unity-based 2D, 3D, and VR/AR environments using various AI methods. It provides Python APIs for training and Unity C# scripts for environment simulation. With over 17 example Unity environments, it is well-suited for evaluating XR testing approaches, including agent-based testing Andrade et al. 2023

Clumsy[34] and **Wireshark**[35] are tools for network simulation and analysis. Both were used in PS15 to simulate network- and application-based attacks. These tools are applicable to networked applications, including XR clients and servers, enabling the evaluation of resilience and performance under adverse network conditions.

6.2.4 Research tools

This section examines research tools specifically designed for XR testing, excluding general tools for tasks like data analysis. We assess each tool's source (primary studies or external references), key functionalities, supported platforms, and availability. This analysis is based on publicly available versions, focusing on implementations rather than techniques reported in the papers. While we did not run the tools, we thoroughly reviewed their documentation and repositories. Table 15 lists the tools analysed.

iv4XR[36] is a suite of tools for automated testing for XR applications. It includes frameworks for agent-, model-, and reinforcement learning-based testing, as well as user experience testing.

ARCHIE[37] is a Unity Editor plugin for usability testing in mobile and wearable AR apps. The repository includes Unity-based examples and supports cloud functions.

MAR-Security[38] is a framework for preventing hidden operations in mobile AR apps. Its repository includes an Android project implementing the detection mechanism and scripts for collecting runtime data from Android devices.

StereoID[39] is a tool for detecting stereoscopic visual inconsistencies linked to cybersickness. However, the tool is not currently accessible.

31 https://airtest.netease.com/

32 https://alttester.com/tools/

33 https://github.com/Unity-Technologies/ml-agents

34 https://jagt.github.io/clumsy/

35 https://www.wireshark.org/

36 https://github.com/iv4xr-project

37 https://github.com/lehmansarahm/ARCHIE

38 https://github.com/lehmansarahm/MAR-Security

39 https://sites.google.com/view/stereoid

Table 15 Research tools in the primary studies

Name	Source	Function	Platform	Avail.
iv4xr	PS3	Agent-based testing	N/A	T
ARCHIE	PS7	Usability testing	Unity	T
MAR-Security	PS18	Hidden operation detection	Android	T
StereoID	PS19	Cybersickness detection	N/A	F
PredART	PS23	Object misplacement prediction	Unity	T
VOPA	PS27	Object misplacement assessment	N/A	F
VRGuide	PS32	VR scene exploration	Unity	T
VRTest	PS33	VR scene exploration	Unity	T
AutoQuest	Herbold and Harms (2013)	Usability smell detection	N/A	F
TESTAR	Vos et al. (2021)	Scriptless GUI testing	desktop, web, mobile	T

PredART[40] includes two types of scripts: a C# camera control script for Unity projects, and scripts for machine learning model implementation and training.

VOPA[41], is a tool designed to assess virtual object misplacement. However, it is not currently publicly accessible.

VRGuide[42] and **VRTest**[43] are automated VR testing tools for scene exploration. While each tool employs different exploration strategies, both provide Unity scripts for their implementations.

AutoQUEST (Herbold and Harms 2013) detects usability smell by analysing recorded user data. While its website[44] is accessible, not the source code but compiled Java (.jar) files are available.

TESTAR[45] (Vos et al. 2021) is an open-source tool for scriptless automated testing of desktop, web and mobile apps at the GUI level. The repository includes documentation for setup and execution. PS24 (Pastor Ricós 2022) references it as a tool that can extend for scriptless testing in XR environments.

6.3 Implications for practitioners

Based on our analysis of datasets and tools referenced in the primary studies, our mapping study reveals two useful insights for XR practitioners.

First, regarding tool selection guidance, Table 14 provides a curated selection of industrial tools organised by platform and testing task. While our findings are based on the primary studies selected for this mapping study, we acknowledge that additional options like Meta XR Simulator (discussed in § 1) may also be valuable for certain testing scenarios.

40 https://sites.google.com/view/predart2022

41 https://sites.google.com/view/vopa-for-artesting/home

42 https://sites.google.com/view/vrguide2023

43 https://sites.google.com/view/vrtest2021

44 https://autoquest.informatik.uni-goettingen.de/trac/wiki

45 https://testar.org/, https://github.com/TESTARtool/TESTAR_dev

Second, concerning research-to-practice opportunities, Table 15 highlights the research tools that address gaps in current industrial offerings. Though these may require additional implementation effort, they provide cutting-edge capabilities for organisations with specialised testing needs or those seeking competitive advantages in XR quality assurance.

6.4 Challenges and future research directions

This section explores the open issues and potential future research directions based on the findings of this mapping study.

During the study selection process (cf. §4.2.1), some studies are excluded as they do not directly align with the focus on testing-related research. However, these studies address challenges that could inspire novel testing approaches by being adapted to specific XR testing needs. By integrating insights from these excluded studies with the findings from our mapping study, we aim to present meaningful and actionable future research directions to advance XR software testing.

6.4.1 Interaction formalisation

As discussed in §5.2.2, *user interaction* is the most common testing test objective for functional testing, indicating the importance of interaction testing in XR apps. In §5.1.2, we classified *scene testing* studies that validate XR functionality through interactions with virtual objects and scene navigation. However, these approaches provide limited context on the specific interactions required to trigger objects (e.g., touching) or complete navigation tasks (e.g., reaching a destination).

Unlike 2D GUI apps, where interaction types are relatively straightforward, XR apps' 6DOF nature demands more diverse interaction types. Moreover, XR interaction methods may vary based on the deployment platform and device capabilities.

Drawing from prior research, formal gesture descriptions have proven effective in automating UI testing for mobile apps (Hesenius et al. 2014) and could similarly benefit XR apps. However, this requires a predefined set of XR-specific interaction types, which remains an open challenge (Börsting et al. 2022). Leveraging these predefined interactions could support cross-device compatibility testing and facilitate the development of reusable testing frameworks for diverse XR platforms.

We recommend systematic empirical studies to categorise XR-specific interactions (e.g., gestures, haptic feedback) by analysing documentation from XR development platforms and open-source projects to create standardised interaction taxonomies.

6.4.2 Test oracle automation

In §5.2.1, we identified *test automation*, *test input generation*, and *test generation* as the most frequent test activities for XR apps. Among these, *test generation*– which involves generating both test inputs and oracles–remains the least explored. Non-crashing functional bugs often require manual validation, with current approaches focusing primarily on crash bugs due to the lack of automated oracles (Su et al. 2021).

Automating oracles is crucial for overcoming this bottleneck and advancing automated testing (Barr et al. 20215).

While some research has addressed the oracle problem for XR apps, the specific oracles needed to validate functionality remain unclear and vary by system (Pastor Ricós 2022). For example, detecting collision and object misplacement may require distinct oracles, each demanding tailored techniques. Addressing this gap necessitates a deeper understanding of the problem and the development of novel solutions.

We propose (1) investigating which XR app characteristics can serve as reliable test oracles, and (2) determining the most effective oracle types (e.g., assertions, contracts, or metamorphic relationships (Molina et al. 2025)) for different XR testing scenarios.

6.4.3 XR-specific testing

XR-specific requirements encompass a wide range of test targets, including real-time collision and occlusion, as well as key AR features such as tracking and registration (Doerner et al. 2022). §5.1.2 identifies *XR-specific testing* as a primary research focus in XR software testing.

Additionally, studies identified during the selection process provide insights into testing these requirements. For example, several studies (Cheng and Qu 2021; Wei and Xinxin 2012; Xu and Sun 2023; Jin et al. 2021; Zhang et al. 2014) propose effective collision testing techniques. However, these studies mainly focus on experimental simulations and have not been applied to specific XR apps. Their methodologies could be adapted to enable systematic collision testing in XR apps, such as instrumenting specific objectives in an XR app to yield collision information.

The unique nature of XR-specific requirements calls for novel testing methodologies not present in other software domains, underscoring the need for tailored approaches and further research. For effective testing, we recommend first conducting systematic studies to analyse the software manifestation of XR-specific features. This analysis should identify observable behaviours in XR apps and determine which software testing techniques would be most effective for validating these unique characteristics. Such foundational work is essential before developing specialised XR testing methodologies.

6.4.4 Software-centric usability testing

Cybersickness is the most common usability issue in XR apps, with several studies proposing software-centric techniques for automated detection, as detailed in §5.2.2. Additionally, many user-centric studies explore the nature of cybersickness (§3.1), providing a foundation for developing software-centric detection methods.

Beyond cybersickness, we identified usability-focused user studies during the study selection process that could inform automated testing techniques from a software-centric perspective. For example, Kia et al. (2023) highlighted factors affecting users' muscular loads during AR app interactions, such as interaction error rates and target size. These factors could be formalised into software models to automate the detection of similar usability issues, addressing a broader range of challenges in XR app usability.

To bridge the gap between software- and user-centric approaches, we recommend integrating findings from user studies into automated testing frameworks. This integration would enable the detection of common usability issues without requiring human evaluation, making usability testing more scalable and consistent across XR apps.

6.4.5 AI for XR testing

Advancements in AI, particularly large language models (LLMs) and reinforcement learning techniques, present opportunities to enhance XR app testing.

As discussed in §6.4.2, test oracle automation remains a significant challenge in XR testing. While crowdsourcing has been shown to effectively address oracle-related tasks (Pastore et al. 2013; Rafi et al. 2023), recent progress in LLMs offers a potential alternative for automating text-based tasks (Thomas et al. 2024), which benefits the generation of human-readable assertions, validating expected outputs, and synthesizing test expectations from natural language specifications.

The oracle problem in XR systems is complex due to their reliance on 3D graphics. However, multimodal LLMs, which process both textual and visual information, have demonstrated capabilities in understanding graphical content, ranging from 2D screenshots to 3D assets (Liu et al. 2024; Qiu et al. 2024). These advancements could enable more robust testing of intricate graphics systems, including XR apps.

Furthermore, LLMs have been effectively used to generate unit tests for Unity-based game development (Paduraru et al. 2024). Given the shared Unity platform, these techniques could potentially be adapted for XR app unit testing, further advancing automation in this domain.

In addition, deep reinforcement learning and imitation learning techniques have demonstrated capabilities to both play (complete specific tasks) and test (explore unknown scenarios) video games (Zheng et al. 2019). We suggest leveraging these techniques to tackle the interactive challenges of XR app testing.

7 Conclusion

This paper presents the methodologies, results, and findings of a systematic mapping study on software testing for XR applications. From an initial pool of 1167 studies retrieved from a digital library, we selected 34 relevant studies for in-depth analysis.

We classified these studies and extracted meaningful information to address key research questions regarding the current research status, test facets (including test activities, concerns, and techniques), and evaluation methodologies employed in XR testing. Additionally, we catalogued datasets and tools referenced in these studies, offering a valuable resource for researchers and practitioners to build upon and advance their work.

The mapping study identifies several open issues and outlines promising future research directions. Our findings highlight the growing importance of XR testing and provide a foundation for advancing methodologies to address its unique challenges. As XR technology rapidly evolves with new platforms, devices and applications, testing methodologies must not only adapt to support these innovations but also leverage the emerging capabilities they offer. Advanced features and hardware capabilities present both challenges and opportunities for testing. Future testing approaches will need to accommodate the increasing complexity of XR environments and the integration of AI-driven behaviours that characterize next-generation XR systems.

In our future work, we plan to focus on the challenge of interaction formalisation for XR testing. By systematically mapping interactions in XR apps to specific user actions, we aim to develop a comprehensive tool capable of automatically generating user action sequences for executing certain testing tasks. The tool would also maintain traceability of action sequences to facilitate bug analysis and reproduction.

8 Appendix A List of primary studies

The list corresponds to the studies prefaced with "PS" throughout the paper.

ID	Title
PS1 (Jung et al. 2017)	360° Stereo image based VR motion sickness testing system
PS2 (Yang and Zhang 2023)	A Study of User Privacy in Android Mobile AR Apps
PS3 (Prasetya et al. 2021)	An Agent-based Architecture for AI-Enhanced Automated Testing for XR Systems, a Short Paper
PS4 (Tramontana et al. 2022)	An Approach for Model Based Testing of Augmented Reality Applications
PS5 (Corrêa Souza et al. 2018)	An automated functional testing approach for virtual reality applications
PS6 (Li et al. 2020)	An Exploratory Study of Bugs in Extended Reality Applications on the Web
PS7 (Lehman et al. 2023)	ARCHIE++ : A Cloud-Enabled Framework for Conducting AR System Testing in the Wild
PS8 (Kirayeva et al. 2023)	Automated Testing of Functional Requirements for Virtual Reality Applications
PS9 (Bierbaum et al. 2003)	Automated testing of virtual reality application interfaces
PS10 (Harms 2019)	Automated Usability Evaluation of Virtual Reality Applications
PS11 (Leykin and Tuceryan 2004)	Automatic determination of text readability over textured backgrounds for augmented reality systems
PS12 (Gunawan et al. 2023)	Blackbox Testing on Virtual Reality Gamelan Saron Using Equivalence Partition Method
PS13 (Kilger et al. 2021)	Detecting and Preventing Faked Mixed Reality
PS14 (Odeleye et al. 2021)	Detecting framerate-oriented cyber attacks on user experience in virtual reality
PS15 (Valluripally et al. 2023)	Detection of Security and Privacy Attacks Disrupting User Immersive Experience in Virtual Reality Learning Environments
PS16 (Qin and Hassan 2023)	DyTRec: A Dynamic Testing Recommendation tool for Unity-based Virtual Reality Software
PS17 (Andrade et al. 2023)	Exploiting deep reinforcement learning and metamorphic testing to automatically test virtual reality applications
PS18 (Lehman et al. 2022)	Hidden in Plain Sight: Exploring Privacy Risks of Mobile Augmented Reality Applications
PS19 (Li et al. 2024)	Less Cybersickness, Please: Demystifying and Detecting Stereoscopic Visual Inconsistencies in Virtual Reality Apps
PS20 (Sarupuri et al. 2018)	LUTE: A Locomotion Usability Test Environment for Virtual Reality
PS21 (Kim et al. 2017)	Measurement of exceptional motion in VR video contents for VR sickness assessment using deep convolutional autoencoder

PS22 (Sendari et al. 2020)	Performance Analysis of Augmented Reality Based on Vuforia Using 3D Marker Detection
PS23 (Rafi et al. 2023)	PredART: Towards Automatic Oracle Prediction of Object Placements in Augmented Reality Testing
PS24 (Pastor Ricós 2022)	Scriptless Testing for Extended Reality Systems
PS25 (Corrêa et al. 2020)	Software Testing Automation of VR-Based Systems With Haptic Interfaces
PS26 (Minor et al. 2023)	Test automation for augmented reality applications: a development process model and case study
PS27 (Yang et al. 2024)	Towards Automatic Oracle Prediction for AR Testing: Assessing Virtual Object Placement Quality under Real-World Scenes
PS28 (Andrade et al. 2019)	Towards the Systematic Testing of Virtual Reality Programs
PS29 (Andrade et al. 2020)	Understanding VR Software Testing Needs from Stakeholders' Points of View
PS30 (Izuazu et al. 2023)	Unravelling the Black Box: Enhancing Virtual Reality Network Security with Interpretable Deep Learning-Based Intrusion Detection System
PS31 (Rzig et al. 2019)	Virtual Reality (VR) Automated Testing in the Wild: A Case Study on Unity-Based VR Applications
PS32 (Wang et al. 2023)	VRGuide: Efficient Testing of Virtual Reality Scenes via Dynamic Cut Coverage
PS33 (Wang 2022)	VRTest: An Extensible Framework for Automatic Testing of Virtual Reality Scenes
PS34 (Figueira and Gil 2022)	Youkai: A Cross-Platform Framework for Testing VR/AR Apps

Author Contributions R.G: Conceptualization, Methodology, Formal analysis, Investigation, Data Curation, Writing - Original Draft, Visualization, Project administration J.R: Methodology, Formal analysis, Investigation, Data Curation, Writing - Review & Editing, Supervision D.S: Methodology, Formal analysis, Investigation, Data Curation, Writing - Review & Editing, Supervision

Data Availability Data is partially provided within the manuscript and is fully available at https://sites.google.com/view/xr-testing

Declarations

Competing interests The authors declare no competing interests.

References

Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press (2008). https://doi.org/10.1017/CBO9780511809163

Andrade, S.A., Nunes, F.L.S., Delamaro, M.E.: Towards the systematic testing of virtual reality programs. In: 2019 21st symposium on virtual and augmented reality (SVR), pp 196–205 (2019). https://doi.org/10.1109/SVR.2019.00044

Andrade, S.A., Quevedo AJU, Nunes FLS, et al.: Understanding vr software testing needs from stakeholders' points of view. In: Symposium on virtual and augmented reality (SVR), pp. 57–66 (2020). https://doi.org/10.1109/SVR51698.2020.00024

Andrade, S.A., Nunes, F.L.S., Delamaro, M.E.: Exploiting deep reinforcement learning and metamorphic testing to automatically test virtual reality applications. Software Testing, Verification and Reliability 33(8) (2023). https://doi.org/10.1002/stvr.1863

Ardito, L., Coppola, R., Morisio, M, et al.: Espresso vs. eyeautomate: An experiment for the comparison of two generations of android gui testing. In: Proc. of the 23rd Intl. Conf. on evaluation and assessment in software engineering. ACM, EASE '19, p 13–22 (2019). https://doi.org/10.1145/3319008.3319022

Barbieri L, Bruno F, Muzzupappa M.: Virtual museum system evaluation through user studies. J. Cult. Herit. 26:101–108 (2017). https://doi.org/10.1016/j.culher.2017.02.005

Barr, E.T., Harman, M., McMinn, P., et al.: The oracle problem in software testing: A survey. IEEE Trans. Soft. Eng. **41**(5), 507–525 (2015). https://doi.org/10.1109/TSE.2014.2372785

Bierbaum, A., Hartling, P., Cruz-Neira, C.: Automated testing of virtual reality application interfaces. In: Proc. of the Workshop on Virtual Environments 2003. ACM, EGVE '03, p 107–114 (2003). https://doi.org/10.1145/3551349.3561160

Börsting I, Heikamp M, Hesenius M, et al.: Software engineering for augmented reality - a research agenda. Proc ACM Hum-Comput Interact 6 (2022). https://doi.org/10.1145/3532205

Bouvier, P., De Sorbier, F., Chaudeyrac, P., et al.: Cross benefits between virtual reality and games. In: Intl. Conf. and Industry Symposium on Computer Games, Animation, Multimedia, IPTV, Edutainment and Security (CGAT'08) (2008). https://doi.org/10.5176/978-981-08-8227-3_cgat08-26

Breen, D., Rose, E., Whitaker, R.: Interactive occlusion and collision of real and virtual objects in augmented reality. Proc of Eurographics Poitiers, France (2000)

Brown-Johnson C, Berrean B, Cataldo J.: Development and usability evaluation of the mhealth tool for lung cancer (mhealth tlc): A virtual world health game for lung cancer patients. Patient Education and Counseling 98 (2015).https://doi.org/10.1016/j.pec.2014.12.006

Casey, P., Baggili, I., Yarramreddy, A.: Immersive virtual reality attacks and the human joystick. IEEE Transactions on Dependable and Secure Computing **18**(2), 550–562 (2021). https://doi.org/10.1109/TDSC.2019.2907942

Charron, O., Lallement, A., Jarnet, D., et al.: Automatic detection and segmentation of brain metastases on multimodal mr images with a deep convolutional neural network. Comput. Biol. Med. **95**, 43–54 (2018). https://doi.org/10.1016/j.compbiomed.2018.02.004

Chen, C.J., Lau, S.Y., Chuah, K.M., et al.: Group usability testing of virtual reality-based learning environments: A modified approach. Procedia - Soc. Behav. Sci. **97**, 691–699 (2013). https://doi.org/10.1016/j.sbspro.2013.10.289

Cheng, S., Qu, H.: Key Issues of Real-time Collision Detection in Virtual Reality. Intl Journal of Frontiers in Engineering Technology 3(5) (2021). https://doi.org/10.25236/IJFET.2021.030505

Corrêa, C.G., Delamaro, M.E., Chaim, M.L., et al.: Software testing automation of vr-based systems with haptic interfaces. Comput. J. **64**(5), 826–841 (2020). . https://doi.org/10.1093/comjnl/bxaa054

Corrêa Souza, A.C., Nunes, F.L.S., Delamaro, M.E.: An automated functional testing approach for virtual reality applications. Software Testing, Verification and Reliability 28(8) (2018). https://doi.org/10.1002/stvr.1690

Davis, S., Nesbitt, K., Nalivaiko, E.: A systematic review of cybersickness. In: Proc. of the 2014 conference on interactive entertainment. ACM, IE2014, pp. 1–9 (2014). https://doi.org/10.1145/2677758.2677780

Dey, A., Billinghurst, M., Lindeman, R.W., et al.: A Systematic Review of 10 Years of Augmented Reality Usability Studies: 2005 to 2014. Frontiers in Robotics and AI 5 (2018). https://doi.org/10.3389/frobt.2018.00037

Doerner, R., Broll, W., Grimm, P., et al.: (eds) Virtual and Augmented Reality (VR/AR): Foundations and Methods of Extended Realities (XR). Springer Intl. Publishing (2022). https://doi.org/10.1007/978-3-030-79062-2

Emery, V., Jacko, J., Kongnakorn, T., et al.: Identifying critical interaction scenarios for innovative user modeling. In: Proc. of the 1st Intl. Conf. on universal access in human-computer interaction, pp. 481–485 (2001)

Figueira, T., Gil, A.: Youkai: A cross-platform framework for testing vr/ar apps. In: HCI Intl. 2022 – Late Breaking Papers: interacting with extended reality and artificial intelligence. Springer Nature Switzerland, pp. 3–12 (2022). https://doi.org/10.1007/978-3-031-21707-4_1

Garousi, V., Mesbah, A., Betin-Can, A., et al.: A systematic mapping study of web application testing. Inf. Soft. Technol. **55**(8), 1374–1396 (2013). https://doi.org/10.1016/j.infsof.2013.02.006

Geiger, A., Lenz, P., Stiller, C., et al.: Vision meets robotics: The kitti dataset. Intl J. Robot. Res. **32**(11), 1231–1237 (2013). https://doi.org/10.1177/0278364913491297

Gu, R., Rojas, J.M.: An empirical study on the adoption of scripted gui testing for android apps. In: 2023 38th IEEE/ACM international conference on automated software engineering workshops (ASEW), pp. 179–182 (2023). https://doi.org/10.1109/ASEW60602.2023.00030

Herbold, S., Harms, P.: Autoquest – automated quality engineering of event-driven software. In: 2013 IEEE Sixth Intl. Conf. on software testing, verification and validation workshops, pp. 134–139 (2013). https://doi.org/10.1109/ICSTW.2013.23

Harms, P.: Automated usability evaluation of virtual reality applications. ACM Trans Comput-Hum Interact 26(3) (2019). https://doi.org/10.1145/3301423

Hertzum, M.: Usability testing: A practitioner's guide to evaluating the user experience. Synthesis Lectures on Human-Centered Informatics **1**, i–105 (2020). https://doi.org/10.2200/S00987ED1V01Y202001HCI045

Hesenius M, Griebe T, Gries S, et al.: Automating ui tests for mobile applications with formal gesture descriptions. In: Proc. of the 16th Intl. Conf. on human-computer interaction with mobile devices & services. ACM, MobileHCI '14, p 213–222 (2014). https://doi.org/10.1145/2628363.2628391

Hu, Y., Azim, T., Neamtiu, I.: Versatile yet lightweight record-and-replay for android. SIGPLAN Not **50**(10), 349–366 (2015). https://doi.org/10.1145/2858965.2814320

Izuazu, U.U., Kim, D.S., Lee, J.M.: Unravelling the black box: Enhancing virtual reality network security with interpretable deep learning-based intrusion detection system. In: Intl. Conf. on Information and Communication Technology Convergence (ICTC), pp 928–931 (2023). https://doi.org/10.1109/ICTC58733.2023.10392826

Jin, Y., Geng, J., He, Z., et al.: A capsule-based collision detection approach of irregular objects in virtual maintenance. Assembly Autom. **41**(1), 89–105 (2021). https://doi.org/10.1108/AA-12-2019-0224

Jung, S.M., Oh, S.H., Whangbo, T.k.: 360∘ stereo image based vr motion sickness testing system. In: 2017 Intl. Conf. on Emerging Trends & Innovation in ICT (ICEI), pp. 150–153. https://doi.org/10.1109/ETIICT.2017.7977027

Kamińska, D., Zwoliński, G., Laska-Leśniewicz, A.: Usability testing of virtual reality applications-the pilot study. Sensors **22**(4) (2022). https://doi.org/10.3390/s22041342

Kavanagh, S., Luxton-Reilly, A., Wuensche, B., et al.: A systematic review of virtual reality in education. Themes Sci. Technol. Educ. **10**(2), 85–119 (2017)

Kia, K., Hwang, J., Kim, J.H.: Effects of error rates and target sizes on neck and shoulder biomechanical loads during augmented reality interactions. Appl. Ergon. **113**, 104107 (2023). https://doi.org/10.1016/j.apergo.2023.104107

Kilger, F., Kabil, A., Tippmann, V., et al.: Detecting and preventing faked mixed reality. In: 2021 IEEE 4th Intl. Conf. on Multimedia Information Processing and Retrieval (MIPR), pp. 399–405 (2021). https://doi.org/10.1109/MIPR51284.2021.00074

Kim, H.G., Baddar, W.J., Lim, H.t., et al.: Measurement of exceptional motion in vr video contents for vr sickness assessment using deep convolutional autoencoder. In: Proc. of the 23rd ACM Symposium on Virtual Reality Software and Technology. ACM, VRST '17 (2017). https://doi.org/10.1145/3139131.3139137

Kim, Y., Kim, H., Kim, Y.O.: Virtual Reality and Augmented Reality in Plastic Surgery: A Review. Arch. Plast. Surg. **44**(3), 179–187 (2017). https://doi.org/10.5999/aps.2017.44.3.179

Kim, Y.M., Rhiu, I., Yun, M.H.: A systematic review of a virtual reality system from the perspective of user experience. Intl Journal of Human-Computer Interaction **36**(10), 893–910 (2020). https://doi.org/10.1080/10447318.2019.1699746

Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Tech. Rep. EBSE-2007-01, Keele University (2007)

Kong, P., Li, L., Gao, J., et al.: Automated testing of android apps: A systematic literature review. IEEE Trans. Reliab. **68**(1), 45–66 (2019). https://doi.org/10.1109/TR.2018.2865733

Kundu, R.K., Elsaid, O.Y., Calyam, P., et al.: Vr-lens: Super learning-based cybersickness detection and explainable ai-guided deployment in virtual reality. In: Proc. of the 28th Intl. Conf. on Intelligent User Interfaces. ACM, IUI '23, p 819–834 (2023). https://doi.org/10.1145/3581641.3584044

Kirayeva, R.R., Khafizov, M.R., Turdiev, T.T., et al.: Automated testing of functional requirements for virtual reality applications. In: 2023 IEEE XVI Intl. Scientific and Technical Conference Actual Problems of Electronic Instrument Engineering (APEIE), pp 1760–1764 (2023). https://doi.org/10.1109/APEIE59731.2023.10347611

Kuri, M., Karre, S.A., Reddy, Y.R.: Understanding software quality metrics for virtual reality products - a mapping study. In: Proc. of the Innovations in Software Engineering Conference (Formerly Known as India Software Engineering Conference). ACM, ISEC '21 (2021). https://doi.org/10.1145/3452383.3452391

Lam, W., Wu, Z., Li, D., et al.: Record and replay for android: are we there yet in industrial cases? In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, ESEC/FSE 2017, pp. 854–859 (2017). https://doi.org/10.1145/3106237.3117769

Lehman, S.M., Alrumayh, A.S., Kolhe, K., et al.: Hidden in plain sight: Exploring privacy risks of mobile augmented reality applications. ACM Trans Priv Secur **25**(4) (2022). https://doi.org/10.1145/3524020

Lehman, S.M., Elezovikj, S., Ling, H., et al.: Archie++: A cloud-enabled framework for conducting ar system testing in the wild. IEEE Trans. Vis. Comput. Graph. **29**(4), 2102–2116 (2023). https://doi.org/10.1109/TVCG.2022.3141029

Lele, A.: Virtual reality and its military utility. J. Ambient Intell. Humanized Comput. **4**(1), 17–26 (2013). https://doi.org/10.1007/s12652-011-0052-4

Li, S., Wu, Y., Liu, Y., et al.: An exploratory study of bugs in extended reality applications on the web. In: 2020 IEEE 31st Intl. symposium on software reliability engineering (ISSRE), pp. 172–183 (2020). https://doi.org/10.1109/ISSRE5003.2020.00025

Li, S., Gao, C., Zhang, J., et al.: Less cybersickness, please: Demystifying and detecting stereoscopic visual inconsistencies in virtual reality apps. Proc ACM Softw Eng **1**(FSE) (2024). https://doi.org/10.1145/3660803

Leykin, A., Tuceryan, M.: Automatic determination of text readability over textured backgrounds for augmented reality systems. In: IEEE/ACM Intl. Symposium on Mixed and Augmented Reality, pp 224–230 (2004). https://doi.org/10.1109/ISMAR.2004.22

Liu, Z., Li, C., Chen, C., et al.: Vision-driven automated mobile gui testing via multimodal large language model (2024). arXiv:2407.03037

Lu, C., Shi, J., Jia, J.: Abnormal event detection at 150 fps in matlab. In: IEEE Intl. Conf. on Computer Vision, pp. 2720–2727 (2013). https://doi.org/10.1109/ICCV.2013.338

Mahadevan, V., Li, W., Bhalodia, V., et al.: Anomaly detection in crowded scenes. In: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pp. 1975–1981 (2010). https://doi.org/10.1109/CVPR.2010.5539872

Milgram, P., Takemura, H., Utsumi, A., et al.: Augmented reality: A class of displays on the reality-virtuality continuum. Telemanipulator and Telepresence Technologies **2351** (1994). https://doi.org/10.1117/12.197321

Modarressi, S.N., Maurer, F., Wang, N.: Capture and replay testing tool for xr applications. In: RealXR@AVI (2024). https://api.semanticscholar.org/CorpusID:270563554

Minor, S., Ketoma, .K., Meixner, G.: Test automation for augmented reality applications: a development process model and case study. i-com 22(3):175–192 (2023). https://doi.org/10.1515/icom-2023-0029

Molina, F., Gorla, A., d'Amorim, M.: Test oracle automation in the era of llms. ACM Trans Softw Eng Methodol (2025). https://doi.org/10.1145/3715107, just Accepted

Paduraru, C., Stefanescu, A., Jianu, A.: Unit test generation using large language models for unity game development. In: Proc. of the 1st ACM Intl. Workshop on Foundations of Applied Software Engineering for Games. ACM, FaSE4Games 2024, p 7–13 (2024). https://doi.org/10.1145/3663532.3664466

Pastor Ricós, F.: Scriptless testing for extended reality systems. In: Research Challenges in Information Science. Springer Intl. Publishing, pp. 786–794 (2022). https://doi.org/10.1007/978-3-031-05760-1_56

Pastore, F., Mariani, L., Fraser, G.: Crowdoracles: Can the crowd solve the oracle problem? In: 2013 IEEE Sixth Intl. Conf. on Software Testing, Verification and Validation, pp. 342–351 (2013).https://doi.org/10.1109/ICST.2013.13

Odeleye, B., Loukas, G., Heartfield, R., et al.: Detecting framerate-oriented cyber attacks on user experience in virtual reality. In: 1st Intl. Workshop on Security for XR and XR for Security (2021)

Petersen, K., Feldt, R., Mujtaba, S., et al.: Systematic mapping studies in software engineering. In: Proc. of the 12th Intl. Conf. on Evaluation and Assessment in Software Engineering. BCS Learning & Development Ltd., EASE'08, pp. 68–77 (2008)
Petersen, K., Vakkalanka, S., Kuzniarz, L.: Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Soft. Technol. **64**, 1–18 (2015). https://doi.org/10.1016/j.infsof.2015.03.007
Politowski, C., Guéhéneuc, Y.G., Petrillo, F.: Towards automated video game testing: still a long way to go. In: Proc. of the 6th Intl. ICSE Workshop on Games and Software Engineering: Engineering Fun, Inspiration, and Motivation. ACM, GAS '22, p 37–43 (2022). https://doi.org/10.1145/3524494.3527627
Prasetya, I.S.W.B., Shirzadehhajimahmood, S., Ansari, S.G., et al.: An agent-based architecture for ai-enhanced automated testing for xr systems, a short paper. In: Intl. Conf. on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, pp. 213–217 (2021). https://doi.org/10.1109/ICSTW52544.2021.00044
Priem, J., Piwowar, H., Orr, R.: Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts (2022). https://arxiv.org/abs/2205.01833
Qiao, X., Ren, P., Dustdar, S., et al.: Web ar: A promising future for mobile augmented reality-state of the art, challenges, and insights. Proc. of the IEEE **107**(4), 651–666 (2019). https://doi.org/10.1109/JPROC.2019.2895105
Qiu, Z., Liu, W., Feng, H., et al.: Can large language models understand symbolic graphics programs? (2024).https://arxiv.org/abs/2408.08313
Qu, C., Che, X., Ma, S., et al.: Bio-physiological-signals-based vr cybersickness detection. CCF Transactions on Pervasive Computing and Interaction **4** (2022). https://doi.org/10.1007/s42486-022-00103-8
Qin, X., Hassan, F.: Dytrec: A dynamic testing recommendation tool for unity-based virtual reality software. In: Proc. of the 37th IEEE/ACM Intl. Conf. on Automated Software Engineering. ACM, New York, NY, USA (2023). https://doi.org/10.1145/3551349.3560510
Rafi, T., Zhang, X., Wang, X.: Predart: Towards automatic oracle prediction of object placements in augmented reality testing. In: Proc. of the 37th IEEE/ACM Intl. Conf. on automated software engineering. ACM (2023). https://doi.org/10.1145/3551349.3561160
Ramaseri Chandra, A.N., El Jamiy, F., Reza, H.: A review on usability and performance evaluation in virtual reality systems. In: 2019 Intl. Conf. on computational science and computational intelligence (CSCI), pp. 1107–1114 (2019). https://doi.org/10.1109/CSCI49370.2019.00210
Gunawan, R., Wibisono, Y.P., Primasari, C.H., et al.: Blackbox Testing on Virtual Reality Gamelan Saron Using Equivalence Partition Method. Jurnal Buana Informatika 14(01), 11–19 (2023). https://doi.org/10.24002/jbi.v14i01.6606
Robert, S., Kennedy KSBNorman. E., Lane Lilienthal, M.G.: Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. The Intl J. Aviat. Psych. **3**(3), 203–220 (1993). https://doi.org/10.1207/s15327108ijap0303_3
Roberts, J.: The ar/vr technology stack: A central repository of software development libraries, platforms, and tools (2023). https://doi.org/10.48550/arXiv.2305.07842
Rodriguez, I., Wang, X.: An empirical study of open source virtual reality software projects. In: 2017 ACM/IEEE Intl. Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 474–475 (2017). https://doi.org/10.1109/ESEM.2017.65
Rzig, D.E., Iqbal, N., Attisano, I., et al.: Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications. In: Proc. of the 32nd ACM SIGSOFT Intl. symposium on software testing and analysis. ACM, ISSTA 2023, p 1269–1281 (2023). https://doi.org/10.1145/3597926.3598134
Scheibmeir, J., Malaiya, Y.K.: Quality model for testing augmented reality applications. In: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), pp. 0219–0226 (2019). https://doi.org/10.1109/UEMCON47517.2019.8992974
Su, T., Meng, G., Chen, Y., et al.: Guided, stochastic model-based gui testing of android apps. In: Proc. of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE). ACM, pp. 245–256 (2017). https://doi.org/10.1145/3106237.3106298
Su, T., Yan, Y., Wang, J., et al.: Fully automated functional fuzzing of android apps for detecting non-crashing logic bugs. Proc ACM Program Lang 5(OOPSLA) (2021). https://doi.org/10.1145/3485533
Sarupuri, B., Hoermann, S., Whitton, M.C., et al.: Lute: A locomotion usability test environmentfor virtual reality. In: 2018 10th Intl. Conf. on Virtual Worlds and Games for Serious Applications (VS-Games), pp 1–4 (2018). https://doi.org/10.1109/VS-Games.2018.8493432

Tadeja, S., Seshadri, P., Kristensson, P.: Aerovr: An immersive visualisation system for aerospace design and digital twinning in virtual reality. Aeronaut. J. **124**(1280), 1615–1635 (2020). https://doi.org/10.1017/aer.2020.49
Sendari, S., Firmansah, A., Aripriharta.: Performance analysis of augmented reality based on vuforia using 3d marker detection. In: 2020 4th Intl. Conf. on Vocational Education and Training (ICOVET), pp 294–298 (2020). https://doi.org/10.1109/ICOVET50258.2020.9230276
Technologies, U.: Unity - manual: Gameobject (2024). https://docs.unity3d.com/Manual/class-GameObject.html, accessed: 13 Aug 2024
Thomas, P., Spielman, S., Craswell, N., et al.: Large language models can accurately predict searcher preferences. In: Proc. of the 47th Intl. ACM SIGIR conference on research and development in information retrieval. ACM, SIGIR '24, p 1930–1940 (2024). https://doi.org/10.1145/3626772.3657707
Tramontana, P., Amalfitano, D., Amatucci, N., et al.: Automated functional testing of mobile applications: A systematic mapping study. Soft. Qual. J. **27**(1), 149–201 (2019). https://doi.org/10.1007/s11219-018-9418-6
Valluripally, S., Frailey, B., Kruse, B., et al.: Detection of security and privacy attacks disrupting user immersive experience in virtual reality learning environments. IEEE Trans. Serv. Comput. **16**(4), 2559–2574 (2023). https://doi.org/10.1109/TSC.2022.3216539
Vos, T.E.J., Aho, P., Pastor Ricos, F., et al.: testar - scriptless testing through graphical user interface. Soft. Test. Verif. Reliab. **31**(3), e1771 (2021). https://doi.org/10.1002/stvr.1771
Walsh, A.E.: Understanding scene graphs. DrDobb's Journal **27**(7), 17–26 (2002)
Wang, X.: VRTest: An extensible framework for automatic testing of virtual reality scenes. In: Intl. Conf. on software engineering: companion proceedings (ICSE-Companion). ACM, pp. 232–236 (2022). https://doi.org/10.1145/3510454.3516870
Wang, X., Rafi, T., Meng, N.: Vrguide: Efficient testing of virtual reality scenes via dynamic cut coverage. In: 2023 38th IEEE/ACM Intl. Conf. on automated software engineering (ASE). IEEE Computer Society, pp. 951–962 (2023). https://doi.org/10.1109/ASE56229.2023.00197
Washizaki, H. (ed) Guide to the Software Engineering Body of Knowledge (SWEBOK Guide), Version 4.0. IEEE Computer Society (2024) . https://www.swebok.org
Wei, Z., Xinxin, G.: The collision detection algorithm in virtual reality. In: 2012 Intl. Conf. on computer science and electronics engineering, pp. 538–541 (2012). https://doi.org/10.1109/ICCSEE.2012.412
Wieringa, R., Maiden, N., Mead, N., et al.: Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requir Eng **11**, 102–107 (2006). https://doi.org/10.1007/s00766-005-0021-6
Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proc. of the 18th Intl. Conf. on evaluation and assessment in software engineering. ACM, EASE '14 (2014). https://doi.org/10.1145/2601248.2601268
Xu, P., Sun, Q.: Virtual reality collision detection based on improved ant colony algorithm. Appl. Sci. 13(11) (2023). https://doi.org/10.3390/app13116366
Tramontana, P., Luca, M.D., Fasolino, A.R.: An approach for model based testing of augmented reality applications. In: RCIS Workshops (2022)
Yang, A.H.X., Kasabov, N., Cakmak, Y.O.: Machine learning methods for the study of cybersickness: A systematic review. Brain Inf. **9**(1), 24 (2022). https://doi.org/10.1186/s40708-022-00172-6
Yang, X., Zhang, X.: A study of user privacy in android mobile ar apps. in: proc. of the 37th ieee/acm intl. conf. on Automated Software Engineering (ASE'22). ACM (2023). https://doi.org/10.1145/3551349.3560512
Yang, X., Wang, Y., Rafi, T., et al.: Towards automatic oracle prediction for ar testing: Assessing virtual object placement quality under real-world scenes. In: Proc. of the 33rd ACM SIGSOFT Intl. symposium on software testing and analysis. ACM, ISSTA 2024, pp. 717–729 (2024). https://doi.org/10.1145/3650212.3680315
Zein, S., Salleh, N., Grundy, J.: A systematic mapping study of mobile application testing techniques. J. Syst. Sof. **117**, 334–356 (2016). https://doi.org/10.1016/j.jss.2016.03.065
Zhang, M., Zhou, D., Lv, C., et al.: Collision detection technology based on capsule model in virtual maintenance. In: Intl. Conf. on Reliability, Maintainability and Safety (ICRMS), pp. 1150–1155 (2014). https://doi.org/10.1109/ICRMS.2014.7107384

Zhang, X., Tao, J., Tan, K., et al.: Finding critical scenarios for automated driving systems: A systematic mapping study. IEEE Trans. Soft. Eng. **49**(3), 991–1026 (2023). https://doi.org/10.1109/TSE.2022.3170122

Zheng, Y., Xie, X., Su, T., et al.: Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning. In: 2019 34th IEEE/ACM Intl. Conf. on automated software engineering (ASE), pp. 772–784 (2019). https://doi.org/10.1109/ASE.2019.00077