МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки і програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач випускової кафедри

_______Олексій ГОРСЬКИЙ

“____”___________2023 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Тема: “Методика тестування застосунків доповненої реальності”

Виконавець: ____________ Кравець Богдан Олександрович

Керівник: ____________ д.т.н., доцент Чебанюк Олена Вікторівна

Нормоконтролер: __________ к.ф.-м.н., доцент Михайло ОЛЕНІН

КИЇВ 2023

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY
Faculty of cybersecurity and software engineering
Software engineering department

ADMIT TO DEFENCE
Head of the Department
________Oleksiy Gorskyi
“____”__________2023

GRADUATE WORK

(EXPLANATORY NOTE)

GRADUATE OF EDUCATIONAL MASTER’S DEGREE

Topic: “Approach for testing Augmented reality applications”

Performer: _________ Moskalenko Danyil Olegovych

Supervisor: _________ D.Sc, associate professor Chebanyuk OlenaViktorivna

Standard controller: _________ PhD, associate professor Mykhailo OLENIN

KYIV 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки і програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олексій Горський
"___" ________ 2023 р

ЗАВДАННЯ
на виконання дипломної роботи
Москаленка Даниїла Олеговича

1. Тема дипломної роботи: «Методика тестування застосунків доповненої реальності»
затверджена наказом ректора від «29» вересня 2023р. № 1994/ст.

2. Термін виконання роботи: з 02.10.2023 р. до 31.12.2023 р.

3. Вихідні данні до проекту: методологія тестування програмного забезпечення доповненої реальності. Програмний продукт, який буде використовуватись як інструмент, як один із інструментів запропанованої методології.

4. Зміст пояснювальної записки:
   1. Доменний аналіз предметної області - доповненої реальності, методів і методологій тестування, інструментів тестування програм доповненої реальності.
   2. Розгляд запропонованої методології тестування.
   3. Системні обмеження і структура взаємодії засобів для виконання запропонованої методології тестування.
   4. Модель взаємодії програмних засобів для проведення тестування
   5. Вимоги до інструменту тестування.
   6. Структура інструменту.
   7. Робочий прототип інструменту.
   8. Результати застосування методології

5. Перелік ілюстративного матеріалу:
   1. Тема, об’єкт дослідження, предмет дослідження, методи дослідження, гіпотеза.
   2. Опис запропонованої методики.

3. .Складності реалізаці методики
4. Можливі способи реалізаці
5. Застосування запропонованої методики на практиці.
6. Висновки.

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка плану роботи, назви розділів ПЗ та затвердження їх керівником (див.2 лист шаблону)	02.10.23 – 08.10.23	
2.	Написання розділу 1 та допоміжних сторінок, презентація науковому керівнику	08.10.23 – 17.10.23	
3.	Написання розділу 2 і допоміжних сторінок. презентація науковому керівнику.	17.10.23 – 20.10.23	
4.	Перший нормо-контроль 1-2 розділ	15.10.23 – 22.10.23	
5.	Написання розділу 3 і допоміжних сторінок. презентація науковому керівнику	22.10.23 – 18.11.23	
6.	Написання розділу 4 і допоміжних сторінок. презентація науковому керівнику	18.11.23 – 30.11.23	
7.	Загальне редагування пояснювальної записки, графічного матеріалу.	30.11.23 – 10.12.23	
9.	Завершення написання ПЗ. Проходження нормоконтролю. Друк ПЗ Отримання відгуку керівника. Підготовка презентації та доповіді на перед захист.	04.12.23 – 15.12.23	
10.	Передзахист каліф. Роботи. Отримання рецензії	15.12.23 – 17.12.23	
11.	Підготовка документів до захисту та здача їх секретарю ДЕК	18.12.23 – 24.12.23	
12.	Захист кваліф. роботи	27.12.23	

7. Дата видачі завдання 05.09.2022 р.

Керівник: _____________ д.т.н., доцент Олена ЧЕБАНЮК

Завдання прийняв до виконання: _____________ Богдан КРАВЕЦЬ

NATIONAL AVIATION UNIVERSITY

Faculty of cybersecurity and software engineering
Department Software Engineering
Education degree master
Speciality 121 Software engineering
Educational-professional program Software engineering

APPROVED
Head of the Department
______Oleksiy GORSKYI
"___" ________ 2023

Task

on executing the graduation work
Moskalenko Danyil Olegovych

1. Topic of the graduation work: «Approach for testing Augmented reality applications»
   Approved by the order of the rector from 29.09.2023 № 1994/ст.

2. Terms of work execution: from 02.10.2023 to 31.12.2023

3. Source data of the work: develop an approach that will allow solving the task of testing augmented reality software. A software product that will be used as a tool in the methodology.

4. Content of the explanatory note:
   1. Domain analysis of the subject area - augmented reality, testing methods and methodologies, tools for testing augmented reality programs. Proposed approach of testing in the AR development.
   2. Consideration of the proposed testing methodology.
   3. System limitations and the structure of the interaction of means for execution the proposed testing methodology.
   4. Model of software interaction for testing.
   5. Requirements for the testing tool.
   6. Structure of the tool.
   7. Working prototype of the tool.
   8. Results of methodology application

5. List of presentation mandatory slides:

   1. Topic, research object, research subject, research methods, hypothesis.
   2. Description of the proposed approach.
   3. Difficulties in implementing the approach.

4. Possible methods of implementation.
5. Application of the proposed approach in practice.
6. Conclusions.

6. Calendar schedule

№	Task	Execution term	Execution mark
1.	Development of a work plan, names of software sections and their approval by the manager	02.10.23 – 08.10.23	
2.	Writing section 1 and supporting pages, presentation to the supervisor.	08.10.23 – 17.10.23	
3.	Writing section 2 and supporting pages. presentation to the supervisor.	17.10.23 – 22.10.23	
4.	The first norm-control 1-2 section	15.10.23 – 22.10.23	
5.	Writing section 3 and supporting pages. presentation to the supervisor.	22.10.23 – 18.11.23	
6.	Writing section 4 and supporting pages. presentation to the supervisor.	18.11.23 – 30.11.23	
7.	General editing of the explanatory note, graphic material.	30.11.23 – 10.12.23	
9.	Completion of software writing. Passing control norms. Printing software. Receiving feedback from the manager. Preparation of presentations and reports for defense.	04.12.23 – 15.12.23	
10.	Pre-defense of qualifying work. Receiving a review	15.12.23 – 17.12.23	
11.	Preparation of materials for transfer to the secretary of the DEC (software, CD-R with electronic copies of the software, presentations, feedback from supervisor, review, certificate of success, folder: check with the secretary of the DEC)	18.12.23 – 24.12.23	
12.	Graduation work defense	25.12.23 – 31.12.23	

7. Date of issue of the assignment 05.09.2022.

Supervisor: _____________ Olena CHEBANYUK

Task accepted for execution: _____________ Danyil MOSKALENKO

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Підхід до тестування додатків доповненої реальності»: 84 с., рис., табл., джерела інформації.

Об’єкт дослідження – процеси тестування програмного забезпечення доповненої реальності.

Предмет дослідження – методи та засоби тестування програм доповненої реальності, спрямовані на ефективну перевірку їх працездатності під час тестування.

Мета даної роботи - запропонувати та дослідити новий метод тестування програм доповненої реальності, який міг би доповнити існуючі методи та інструменти тестування доповненої реальності. Крім того, у майбутньому це може бути вдосконалено завдяки розробці хмарних технологій доповненої реальності.

Гіпотеза - можливість використання застосунку допвненої реальності тестування іншого застосунку доповненої реальності.

Методи дослідження:

Евристичний метод використовується для виявлення проблем і обмежень, властивих взаємодії між двома додатками AR. Це має вирішальне значення не тільки для тестування однієї з цих програм, але й для порівняння встановлених методів із нещодавно запропонованим.

Метод моніторингу відповідає за відстеження передачі даних від пристрою до емулятора.

Метод аналізу використовується для комплексного вивчення предметної області, домену та відповідної літератури.

Метод синтезу використовується для об’єднання ідей і формування консолідованої думки та висновку на основі проаналізованої літератури.

Метод моделювання допомагає сформулювати гіпотезу щодо функціонування запропонованого методу тестування. Він заглиблюється в роботу та взаємодію його компонентів, проливаючи світло як на переваги, так і на недоліки самого методу, а також на архітектуру взаємодії між інструментами.

Метод моделювання орієнтований на створення середовища за допомогою запропонованого методу тестування. Це змодельоване середовище є ключовим для подальшої перевірки програми, що тестується.

Експериментальний метод використовується для перевірки запропонованого методу тестування та вивчення взаємодії між основними компонентами. Результати магістерської роботи можуть бути використані при розробці та тестуванні додатків доповненої реальності. Вони також можуть сприяти подальшому вдосконаленню методології та, певною мірою, допомогти в концептуалізації хмарних додатків доповненої реальності.

Дослідження та розробки проводилися в операційних системах Windows 10/Windows 11 з використанням мультиплатформенного інструменту Unity, середовища розробки Visual Studio 2022 і редактора Visual Studio Code. Використаною мовою програмування була C#.

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ДОДАТКОВА РЕАЛЬНІСТЬ, ЕМУЛЯТОР, ПОТОКОВА ПЕРЕДАЧА ДАНИХ, ВЗАЄМОДІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, АПАРАТНЕ ОБМЕЖЕННЯ

ABSTRACT

Explanatory note to the thesis "Approach for testing Augmented reality applications": 84 p., fig., tables., information sources.

The object of research - the processes of testing augmented reality software.

The subject of research - the methods and tools for testing augmented reality programs, aimed at effectively verifying their functionality during testing.

The purpose of this work - to propose and investigate a new method for testing augmented reality programs, which could complement existing testing methods and tools for augmented reality. Furthermore, it could be further enhanced with the development of augmented reality cloud technology in the future.

Hypothesis - "the possibility of using an augmented reality application for testing another augmented reality application"

Research methods:

The heuristic method is utilized to identify problems and limitations inherent in the interaction between two AR applications. It is crucial not only for testing one of these applications but also for comparing established methods with the newly proposed one.

The monitoring method is responsible for tracking the data transfer from the device to the emulator.

The analysis method is employed for a comprehensive examination of the subject area, domain, and relevant literature.

The synthesis method is used to amalgamate insights and form a consolidated opinion and conclusion based on the analyzed literature.

The modeling method aids in formulating a hypothesis concerning the functioning of the proposed testing method. It delves into the operation and interaction of its components, shedding light on both the advantages and disadvantages of the method itself, as well as the architecture of the interaction among tools.

The simulation method is focused on creating an environment using the proposed testing method. This simulated environment is pivotal for the subsequent verification of the program under test.

The experimental method is leveraged to validate the proposed testing method and to examine the interaction between essential component parts.

The results of the master's thesis can be used in the development and testing of augmented reality applications. They can also inform further refinement of the methodology and, to some extent, aid in the conceptualization of cloud-based augmented reality applications.

Research and development were conducted on the Windows 10/Windows 11 operating systems, using the Unity multi-platform tool, the Visual Studio 2022 development environment, and the Visual Studio Code editor. The programming language employed was C#.

SOFTWARE ENGINEERING TESTING, ADDITIONAL REALITY, EMULATOR, DATA STREAMING, SOFTWARE INTERACTION, HARDWARE LIMITATION.

TABLE OF CONTENTS

LIST OF ACRONYMS AND ABBREVIATIONS

XR – Extended reality

AR – Augmented reality

VR – Virtual reality

OC – Operating system

INTRODUCTION

Nowadays, there is a significant development of computer technologies, as well as software solutions for them. In addition to the development of conventional programs, there is also the development of the direction of mixed reality (XR), especially augmented reality and virtual reality.

Although some devices for full reality are being developed or are not available to a wide audience. The significant development of mobile has created the concept of using augmented reality applications through personal mobile devices.[MOBILE INDOOR AUGMENTED REALITY. Exploring Applications in Hospitality Environments] In this direction, augmented reality (AR) relies on combining and superimposing virtual information over the real world, providing the user with extra (even real time) computer-based information.

In general, Augmented reality can be described as an enhanced, interactive rendition of the real world, achieved through the incorporation of digital visual elements, sounds, and other sensory stimuli using holographic technology. AR encompasses three key features: the merging of digital and physical realms, real-time interactions, and precise 3D or 2D identification of both virtual and real-world objects.

Now the technology of augmented reality continues to evolve alongside broader technological advancements. However, it also faces certain limitations and risks due to the ongoing development of the field. It has not yet fully realized its potential conceptually. Currently, the proliferation of high-definition cameras, integrated compasses, and inertial systems in mobile devices has created a fertile technological landscape for the development of mobile AR services.

Along with the development of technology and the increase in the number of available devices, the complexity of the software being developed increases, which leads to more potential bugs in the software (software). In addition, software errors can be joined by both errors of new devices for which an augmented reality program can be developed, as well as specific nuances of already existing systems and emulators for them.

Based on the above, the testing process during the development of programs can play a wider role in the creation of augmented reality systems than in the development processes of other types of applications.

Usually, the testing process refers to the process of identifying flaws in developed systems, which often use debugging tools and work in stable operating systems on widely used hardware. However, augmented reality systems can be developed both within the framework of standard hardware with a stable OS, and within the framework of experimental devices and OS. Moreover, it is possible to use various sensors that may not be calibrated. Also, in the testing process, not only bugs in the developed system, but also in the OS or hardware may be found.

CHAPTER 1
DOMAIN ANALYSIS OF THE AUGMENTED REALITY SOFTWARE TESTING AND

1.1. Domain analysis of the argument reality

Augmented Reality (AR) Augmented Reality (AR) is an immersive technology that superimposes digital information, virtual objects, or computer-generated sensory elements onto the real world. This enriches the user's perception of their surroundings, acting as a bridge between the physical and digital realms. AR applications are designed for use in real-world settings, offering users a seamless blend of physical and virtual experiences.

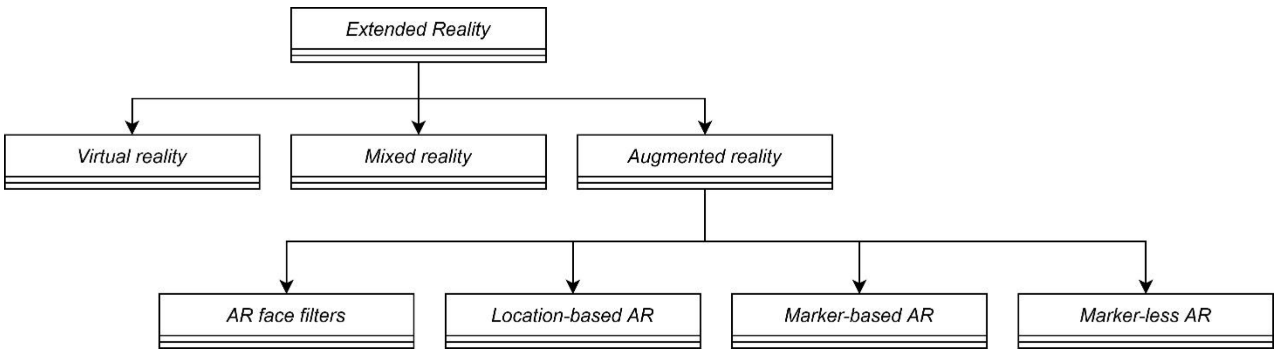

Fig 1.1. External Reality class diagram

AR is part of the broader field known as Extended Reality (XR), which also includes other immersive technologies such as:

-Virtual Reality (VR): Provides immersive experiences that isolate users from the real world, typically achieved through specialized headsets and headphones.

-Mixed Reality (MR): A fusion of AR and VR elements, allowing digital objects to interact with the real world and enabling the integration of virtual elements into genuine environments.

The concept of "augmented reality" is not new; its roots can be traced back to the 1960s with Ivan Sutherland's design of a head-mounted display tracked by mechanical and ultrasonic trackers. However, the term, as it is known today, gained widespread usage starting in 1992 with the work of Caudell and Mizell.

1.2. Main Components of Augmented Reality

Augmented reality technology operates on various devices and consists of five key components, which are essential for its effective functioning:

- Artificial Intelligence (AI): AI allows users to perform actions using voice commands and assists in processing information for AR applications.
- AR Software: These tools and applications enable access to and utilization of
AR. Some businesses develop their own custom AR software.
- Processing Power: AR technology requires substantial processing power, often leveraging the internal operating system of the user's device.
- Lenses: High-quality lenses or image platforms are essential for viewing AR
content. The higher the screen quality, the more realistic the displayed images appear.
- Sensors: AR systems use sensors to collect environmental data, facilitating the alignment of real and digital worlds. This data, captured by cameras, is processed through software to provide a seamless AR experience.

These components collectively enable AR to deliver engaging and interactive experiences. AR has applications across various industries, including entertainment, gaming, education, healthcare, and more. As technology evolves, AR is becoming increasingly integrated into our daily lives, offering new possibilities for enhancing our interactions with the world around us.

1.2.1. Types of augmented reality

There are four primary types of augmented reality (see fig 1.1) : marker-based, marker-less. **AR face filters, Location-based AR** The choice between these types of

AR fundamentally influences how you can display images and information within your AR application.

Marker-Based AR: Marker-based AR relies on image recognition to identify pre-programmed objects within your AR device or application. By placing these objects as reference points within the user's field of view, the AR device can ascertain the camera's position and orientation. Typically, this is achieved by switching the camera to grayscale mode and then using image recognition algorithms to detect a specific marker, comparing it with others stored in its database. Once a match is found, the device uses this data to mathematically determine the object's pose and accurately position the AR image within the real-world environment.

Marker-Less AR: Marker-less AR is a more intricate form of augmented reality, as it doesn't rely on predefined markers or reference points. Instead, it must recognize objects and features as they naturally appear in the user's view. This process involves the use of recognition algorithms that analyze colors, patterns, and similar visual cues to identify objects within the environment. Subsequently, the device utilizes data from various sensors, including time, accelerometers, GPS, and compass information, to orient itself and overlay digital images or information onto the real-world surroundings captured by the camera.

AR face filters involve augmenting a user's face in real time with various digital effects, such as masks, animations, or virtual makeup. These effects track the user's facial features and movements using facial recognition technology, enhancing or transforming their appearance in live video feeds, often for entertainment or social media purposes**.**

Location-based AR leverages GPS and location data to overlay digital content on the user's physical surroundings. By determining the user's real-world location, this type of AR can provide location-specific information, such as nearby points of interest, directions, or geolocated experiences. Location-based AR enhances the user's understanding of their environment and can be used for navigation, tourism, and contextual information delivery.

Each type of AR has its own set of advantages and limitations, and the choice between marker-based and marker-less AR depends on the specific requirements and goals of your AR application. Marker-based AR is often more precise and predictable since it relies on predefined markers, while marker-less AR offers a more flexible and natural interaction with the real world but can be more computationally intensive due to the need for continuous recognition and tracking of objects.

t's worth noting that augmented reality (AR) programs can be categorized into two main types based on their operating environments: those that function in a closed environment and those designed for an open environment.

Closed Environment AR: These AR programs are designed to operate within controlled or confined settings. They often rely on predefined markers, objects, or features that are specific to the closed environment. This approach allows for more precise and predictable AR interactions within a controlled space. Examples of closed environment AR applications include indoor navigation systems within a shopping mall, educational AR experiences within a classroom, or maintenance assistance tools in a factory.

Open Environment AR: On the other hand, open environment AR programs are intended to function in dynamic and uncontrolled surroundings. They are engineered to recognize and interact with objects and features as they naturally occur in the real world. This type of AR requires advanced computer vision and recognition algorithms to identify and track objects and surfaces in real-time. Open environment AR is well-suited for outdoor navigation, tourism applications, and interactive experiences that span various locations.

Closed environment AR offers a high degree of precision but is limited to specific, predefined areas. In contrast, open environment AR provides greater flexibility but demands more complex algorithms and sensors to adapt to diverse and ever-changing surroundings. The decision should align with the desired user experience and the intended application context.

1.2.2 Critical Aspects of Augmented Reality

Augmented reality, as an actively developing field, encompasses several critical aspects that significantly impact its effectiveness and usability:

- Lack of AR Design & Development Standards: A major challenge in the AR industry is the absence of universal standards, leading to software and hardware limitations, difficulties in support, and project testing.
- Security & Privacy Concerns: Inconsistencies in AR programming and negligence raise security and privacy issues. The lack of clear regulations allows for potential misuse and risks such as data leakage, dissemination of unreliable information, and physical harm due to improperly placed virtual objects.
- Technical Limitations: AR requires sophisticated hardware and software, including processors, sensors, cameras, displays, and network capabilities. Inaccuracies in GPS sensors or other components can lead to erroneous display of information. Integration challenges with VR technologies and limited interoperability further hinder broader adoption.
- Limited Interactivity: Compared to VR, AR's interactivity is constrained by its reliance on real-world environments, limiting the extent to which users can manipulate virtual elements.
- Occlusion Issues: A significant challenge in AR development is occlusion, where objects in the environment block the view of virtual elements, requiring substantial processing power for accurate real-time tracking and rendering.
- Challenges in Accurate Tracking: Accurate object tracking is hampered by varying lighting conditions and viewing angles, introducing complexities in the tracking algorithms and potential latency issues.
- Voice Recognition & Processing Limitations: Effective voice recognition depends on robust hardware and specialized algorithms, which can be affected by environmental factors.
- Network Bandwidth & Latency: As AR becomes more widespread, increased demand on network infrastructure can lead to bandwidth constraints and latency issues, impacting application performance.

- Camera Positioning Challenges: Determining the camera angle and location is complex, as it requires algorithms to interpret the viewpoint and orientation relative to virtual objects. Using markers or surface recognition can aid in this process but may neglect non-static objects.
- Physical Object Recognition and Occlusion: Recognizing physical object boundaries and managing occlusion, where parts of virtual objects are overlapped by physical ones, remains a complex task.
- User Experience (UX) Design: The UX of AR apps is critical and should be intuitive, immersive, and seamless, considering user comfort to avoid issues like motion sickness or eye strain.
- Realism and Immersion: The success of AR depends on the realistic integration of virtual and physical elements, including accurate 3D rendering and appropriate scaling.
- Performance and Latency: Low latency and high performance are essential for real-time interaction and maintaining immersion in the AR environment.
- Stability and Tracking: Accurate tracking and stability of virtual elements in physical spaces are crucial, requiring advanced sensor technologies.
- Content Quality and Relevance: AR content should be engaging, relevant, and provide value to the user, encompassing both visual and informational elements.
- Battery Life and Power Efficiency: Optimizing AR apps for power efficiency is vital, especially for mobile applications, to avoid rapid battery depletion.
- Scalability and Integration: AR applications should be scalable and integrable with various systems and technologies for expanded functionality.
- Market Viability and User Adoption: The success of AR apps hinges on understanding the target market, meeting user needs, and ensuring ease of use.

1.3 Domain analysis of the software testing processes

Software testing is a comprehensive and crucial process within the software development life cycle, aimed at examining and ensuring the quality, functionality, and performance of a software product. It involves both validation and verification to provide an objective view of the software, allowing businesses to understand the risks associated with software implementation. This process includes a variety of techniques, ranging from manual interactions to executing test scripts, to detect bugs, errors, and ensure that the software meets its intended purpose and business logic.

Testing not only prevents bugs and reduces development costs but also improves overall performance. It's essential for maintaining software quality, particularly in the development of mobile applications, where attention to detail in testing is increasing. As a process of comparing expected output with actual output, software testing encompasses all aspects of testing, including software security, reliability, correctness, and quality.

Over time, as applications have become more complex, software testing activities have evolved, introducing new techniques and approaches. A key aspect of software testing is to detect failures so that defects can be resolved, although it's acknowledged that testing cannot guarantee perfect functionality under all conditions. It includes various phases such as test strategy, development, bug management, execution, and more.

The software testing lifecycle (STLC) is a sequence of activities conducted in a systematic and planned manner, aimed at improving product quality. It is a subset of the Software Development Life Cycle (SDLC), and its phases are critical to the overall effectiveness and reliability of software development. This lifecycle ensures that testing is managed effectively, catering to various aspects like scalability, resource usage, and reliability, and thereby plays a vital role in the software industry.

1.3.1 Types of Software Testing

Software testing can be classified into multiple categories based on test objectives, strategies, deliverables, ways, and techniques. It can be further divided into automated and manual methods, along with specific testing techniques like black box and white box testing.

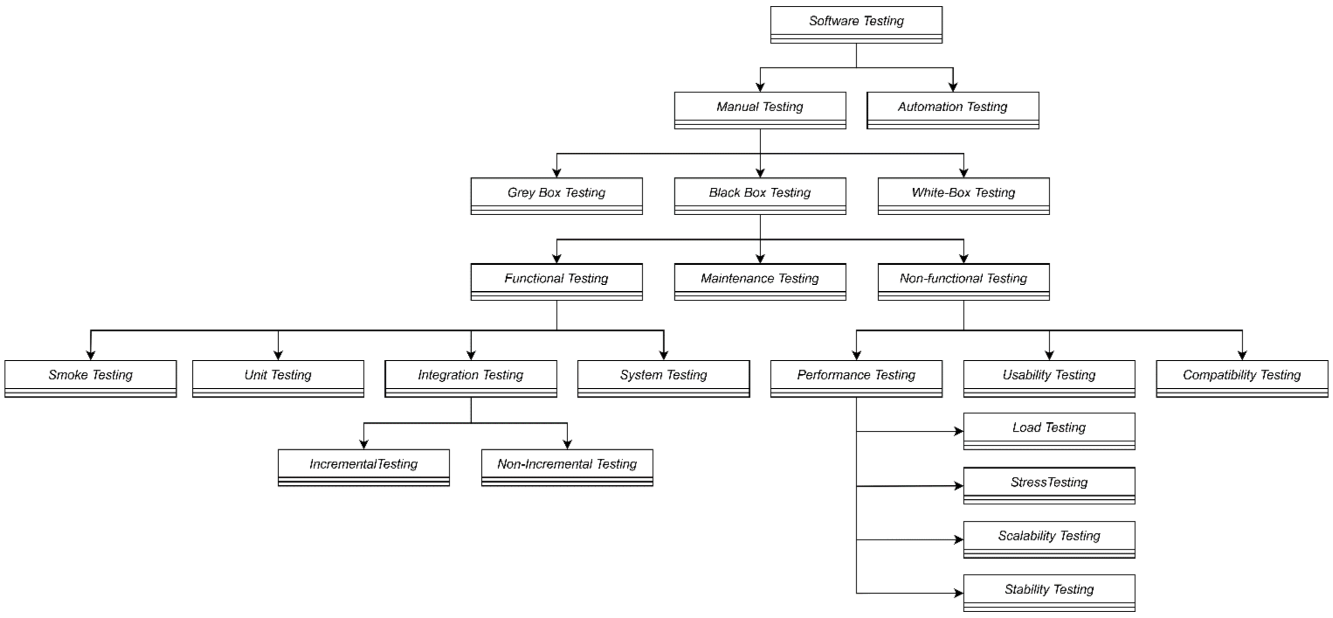

Fig 1.2. Software testing class diagram

Automation Testing: Automation testing, or Test Automation, involves writing scripts and using software to test the product. It is used for re-running manual test scenarios quickly and repeatedly, and for regression, load, performance, and stress testing. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

Manual Testing: Involves testing software manually without using any automation tools or scripts. Testers act as end-users to identify unexpected behaviours or bugs. This method includes various stages like unit integration testing, user acceptance testing etc.

The above categories determine more the resources required for testing than the methods and procedures of their implementation. So, each technique method includes different testing techniques; the most well-known are two techniques: black box testing and white box testing.

Black Box Testing involves testing without access to the source code. Testers focus on the software interface and functionalities, ensuring the program meets project requirements and functions correctly.

White box testing is focuses on the internal structure and logic of a software application. It is also known as "clear box testing," "glass box testing," or "structural testing." The primary goal of white box testing is to ensure that the code and its components work correctly by examining the program's internal workings, code paths, and data flows.

In addition, there exists a less-known category known as Grey Box testing. Grey Box testing requires testers to possess knowledge of the implementation without requiring expertise.

Among these techniques, black box testing is most common. Software testing can be broadly classified into three types:

Functional Testing: It is a type of software testing validates the software's conformance with functional requirements. It checks whether the application functions as specified in the functional requirements. Various types of functional testing include Unit testing, Integration testing, System testing, and Smoke testing.

Unit Testing - testing individual units or components of a software/system to validate that each unit functions as designed. Typically, system programmers and developers perform unit testing.

Integration Testing - combines units and tests them as a group to expose faults in their interactions. It analyzes characteristics such as functional, performance, and reliability requirements imposed on significant design elements.

System Testing - complete, integrated system/software to ensure its compliance with specified requirements.

Smoke testing (build verification testing or sanity testing) is an initial and minimalistic level of software testing performed to verify that the most critical and basic functionalities of a software application are working correctly after a new build or release. The primary purpose of smoke testing is to ensure that the software is stable enough for more extensive testing, such as regression testing or comprehensive functional testing.

Non-functional testing is a type of software testing that assesses the aspects of a software application that do not relate to its specific functionality or features but rather focus on its performance, reliability, scalability, and other quality attributes. These tests evaluate how well the software performs under different conditions and constraints. Non-functional testing is essential to ensure that the software meets user expectations and performs effectively in real-world scenarios. Various types of non-functional testing include Performance testing, Stress testing, and Usability Testing.

Performance Testing: Performance testing evaluates factors like stability, speed, scalability, and responsiveness of an application under specific workloads. It plays a crucial role in ensuring software quality and involves assessing various aspects such as application output, processing speed, data transfer velocity, network bandwidth usage, maximum concurrent users, memory utilization, workload efficiency, and command response times.

Usability Testing: Usability testing involves evaluating a product or service by testing it with representative users, observing their interactions, and noting their feedback.

Stress testing evaluates the behavior of a software application under extreme or unfavorable conditions.

Acceptance testing focuses on evaluating whether a software application meets the specified business requirements and is ready for deployment to end-users or customers.

3. Maintenance testing encompasses modifying and updating the software to meet customer needs. It includes regression testing to ensure that recent code changes do not negatively affect previously functioning parts of the software.

1.4. Testing augmented reality applications

In many cases, AR applications are used on a smartphone, so testing can encompass standard types and methodologies for testing mobile applications by employing testing tools (see table 1.1.).

Table 1.1.

Standard tool for testing applications on smartphones

Tool	Testing type
Selenium	Functional testing
TestComplete	Functional testing, Graphical User Interface testing, Unit testing
Ranorex	Graphical User Interface testing, Compatibility testing
Appium	Graphical User Interface testing, Functional testing
Quick Test Professional	Functional testing, Regression testing
OpenScript	Functional testing, Load testing, Database testing
Janova	Functional testing
Rational Functional	Functional testing, Regression testing, Graphical User Interface testing

Continuation of Table 1.1

However, due to the unique aspects of AR, certain standard tests and tools might not always be practical.

Typically, manual testing is employed for testing AR applications. It often involves two or more individuals to effectively incorporate human factors. In this context, several testing techniques are used, either in their standard form or modified to suit the specific requirements of AR technology. These techniques include functional testing, accessibility testing, usability testing, immersive testing, hardware Testing, Holistic Testing Approach, security testing, loss of connection testing, multiple aspect ratio testing, localization loss, performance testing, compatibility testing.

1.4.1. Special Considerations in AR Testing:

- Integration of AR-specific Factors: While employing both manual and automated testing, special emphasis is placed on AR's unique interaction with real-world environments. This includes testing for spatial awareness, real-world object recognition, and the seamless integration of virtual and physical elements.
- User Experience in AR: Leveraging the principles of usability testing, the focus here extends to the intuitiveness of interacting with augmented elements and the overall immersive experience. This encompasses assessing user comfort, ease of navigation within the AR space, and the responsiveness of AR elements to user actions.

- Hardware Compatibility: Given the diversity of devices on which AR applications can run, hardware testing must ensure optimal performance across various smartphones and AR-specific devices like headsets and wearables.
- Environmental Adaptability: AR applications should be tested in multiple real-world scenarios to evaluate their adaptability to changes in lighting, physical space, and user movements.
- Network Dependency and Connectivity: Special tests are required to assess how AR applications perform under varying network conditions, particularly focusing on scenarios like loss of connectivity to understand the resilience of the application.

Refining Standard Testing Approaches for AR:

- Functional Testing in AR: While the fundamentals of functional testing apply, in AR, this involves ensuring that augmented elements function correctly within their intended real-world contexts.
- Performance Testing with AR Focus: Performance testing should account for the additional processing demands of AR, including real-time rendering of graphics and the handling of complex user interactions.
- Security Testing for AR: The security aspect in AR includes not only data protection but also user privacy concerns, given the technology's interaction with the physical environment and potential access to sensitive information through the device's sensors.

1.4.2. Augmented Reality Applications Testing tools.

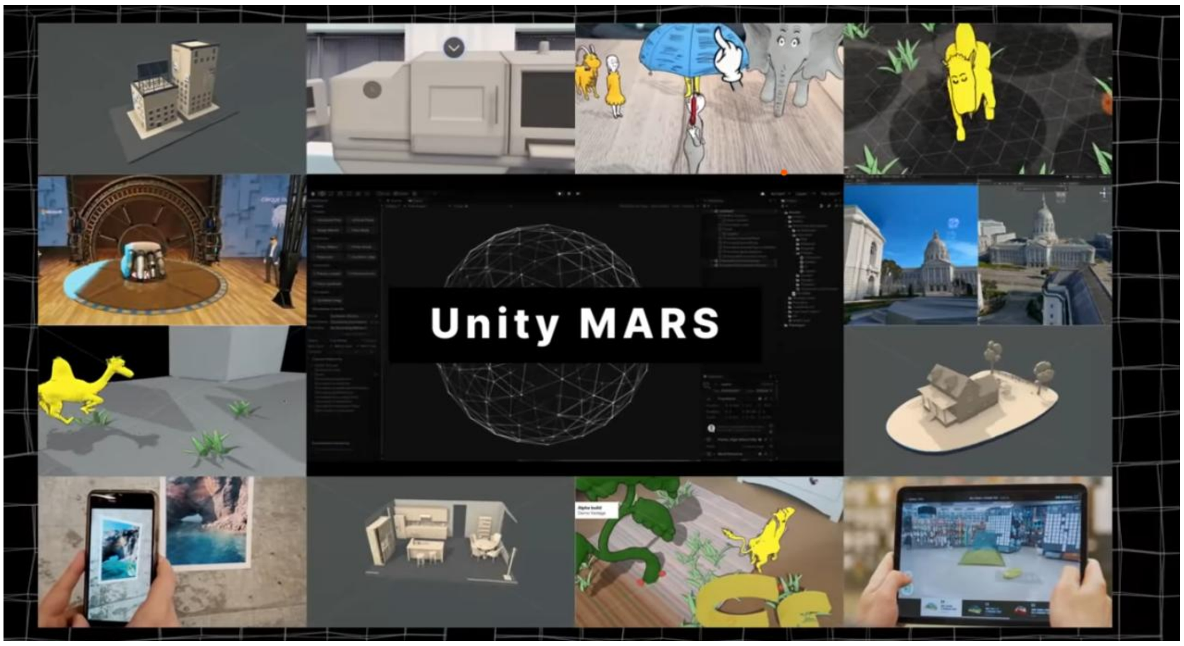

Fig 1.3. Unity MARS presentation

a) Unity Mars: This advanced package of AR tools. It offers templates for developing AR applications, rule-based setups, virtual simulation tools, and high-quality scenes for AR testing. Its cross-platform support accelerates AR application development, allowing testing in a virtual environment without preliminary settings.

However, the annual cost of €552 may be prohibitive for start-ups, and ongoing subscription renewal is necessary due to the difference between free AR tools and Unity Mars tools. A limitation is testing confined to the Unity environment, which may pose challenges for test organizations requiring project source code transfer. There's also a potential for interaction conflicts with third-party AR libraries. It's worth noting that many augmented reality device manufacturers offer complimentary virtual device testing both within and outside the Unity environment, presenting alternative options for developers.

b) Arium is an open-source, lightweight, and extensible framework designed to

streamline the creation of automation tests specifically tailored for XR Applications. It enables scripting for user interaction and object status tracking on stage. The main advantage lies in its capacity to test program components in real scenarios.

However, the manual scripting requirement for each user step, especially in complex interactions, can be cumbersome. Although the program does not account for user-view conditions, additional checks can partially address this issue.

c) GameDriver: GameDriver is a framework that provides developers with an API agent that can be embedded into their program. During development, developers incorporate the GameDriver game object into their game, enabling GameDriver to connect to and control the game while it is running, both during development and in standalone builds. This framework allows communication with the game through a backchannel to the driver's API, enabling the execution of commands asynchronously or synchronously. These commands mimic the input actions of a real user but are executed through a different mechanism. Beyond user input functionality, GameDriver offers features such as logging user code, recording game execution, taking screenshots, recording user input, and accessing application data.

The primary advantage of GameDriver is that all tests are written and executed in a separate, independent application. All operations are performed asynchronously, and connections are established by importing a single package and configuring the required parameters. Consequently, testing can be conducted at any time, independent of the project's state.

However, a major drawback is that GameDriver provides extensive control, potentially introducing vulnerabilities for attackers to exploit or enabling the tracking of user actions.

d) Bitbar: Bitbar is a cloud-based mobile and web application testing platform that supports both live manual app testing and automated testing across various environments. It accommodates testing on desktop browsers (Windows, macOS, Linux) and real iOS and Android devices, offering compatibility with a wide range of modern web browser versions and mobile systems, including Android, iOS, Windows Phone, and Blackberry. Bitbar aims to serve as a comprehensive solution for device and browser testing needs, whether for web, native, or hybrid apps. The platform enables automated testing across multiple devices, supports local testing via SecureTunnel, and provides a customizable app-testing infrastructure to meet specific organizational requirements. Bitbar's scalability and performance capabilities make it a versatile tool for application testing.

Its main advantage lies in providing access to a multitude of devices with varying specifications, essential for AR applications that may perform differently across different hardware. It also facilitates remote testing and automation, speeding up the development cycle and enabling more frequent testing. Testing on actual devices offers a more accurate understanding of how an AR application will perform in real-world scenarios.

However, it's important to note that Bitbar relies on a stable internet connection, and network issues can hinder testing processes. While Bitbar supports a broad range of testing scenarios, it may not offer the same level of customization or specialized tools for AR application testing, which can be more complex due to the integration of real-world environments. Depending on the scale of testing, using a cloud-based platform like Bitbar can be expensive, particularly for small developers or startups. Also remote testing may introduce latency, impacting the testing of AR applications where real-time interaction is crucial.

e) UI Testing Applications: Airtest and XCUITest are testing frameworks primarily focusing on user interface testing. However, they can also be adapted for testing augmented reality (AR) programs utilizing tools like virtual cameras, Azure Spatial Anchor -stores sensor data, video footage with prepared layouts for AR testing. Their main advantage lies in supporting UI automation testing, which is beneficial for AR applications that rely on UI elements overlaid on the real world. These frameworks allow scripting in Python, facilitating the creation of complex test scenarios necessary for AR applications. They employ image recognition technology, enabling interaction with the application by recognizing on-screen elements, a valuable feature in AR where elements can change based on the user's environment and interaction. Moreover, Airtest supports testing on various devices and platforms, which is essential for AR apps designed to function across different hardware and software configurations.

However, these frameworks are not specifically designed for AR, necessitating their use alongside other tools. AR applications often utilize various sensors (e.g., gyroscope, accelerometer), which may not be fully testable without tools like Azure Spatial Anchor. Running complex AR tests can be resource-intensive, potentially leading to performance issues on the testing platform.

f) Performance Metric Tools: These tools provide insights into app performance and include built-in options for Android devices. Examples include OVR Metrics Tools (for analyzing frame rates and thermal values), Logcat (for collecting system logs), Ovrgupprofiler (for accessing GPU pipeline metrics), GPUsystrace (for rendering stage data), RenderDoc (for frame analysis and debugging), and Unity Profiler (for monitoring app performance). Each tool has unique functionalities, but they generally do not have specific disadvantages, except for the potential pre-installation on Android devices.

1.4.3. Evaluation Metrics for Augmented Reality (AR) Applications

Evaluating AR applications presents unique challenges due to the lack of standardized testing methods. However, the evaluation metrics can be broadly categorized into three primary groups: Usability Metrics, User Experience Metrics, and Impact Metrics. Additionally, other relevant metrics, though perhaps less explicit, are also vital in assessment:

a) Usability Metrics:

1) Latency: The delay between user action and system response. Lower latency
is crucial for a seamless AR experience.

2) Accuracy and Precision: How accurately and precisely AR elements are placed in the real world.

3) Frame Rate: The smoothness of the visual display, measured in frames per second.

4) Field of View (FoV): The extent of the observable environment at any

given moment.

5) Object Recognition Time: How quickly the system recognizes and interacts with real-world objects.

b) User Experience Metrics:

1) **User Satisfaction**: Gathered through surveys and interviews, measuring overall satisfaction with the AR experience.

2) **Ease of Use**: Evaluating how intuitive and user-friendly the AR application is.

3) **Engagement**: Assessing how engaging and immersive the AR experience is for users.

4) **Physical Interaction and Ergonomics**: How comfortable and natural it is for users to interact with the AR environment.

c) Impact Metrics:

1) Learning and Performance Improvement**: Assessing whether the AR application helps improve user performance or learning in a given task.**

2) Behavioral Change**: Measuring any changes in user behavior as a result of interacting with the AR application.**

3) Emotional Impact**: Understanding the emotional response elicited by the AR experience.**

d) Additional Metrics:

4) **Battery Consumption**: Important for mobile AR applications, as they can be resource-intensive.

5) **Stability and Robustness**: How well the application performs under different environmental conditions and handling interruptions.

6) **Network Performance**: For AR applications that require internet connectivity, assessing data transfer rates and network latency is crucial.

7) **Rendering Quality**: The visual fidelity of the AR elements, including resolution and textural details.

8) **Privacy and Security**: Especially important given the use of cameras and sensors in public or sensitive environments.

It is also worth noting that as the field of augmented reality matures, there may be a shift toward more standardized metrics and evaluation methodologies. And therefore, these indicators are constantly being improved and adapted so as not to lose relevance. Additionally, depending on the application's use case, additional metrics such as social interaction, collaboration effectiveness, or commercial success may be relevant.

Conclusion

Software testing is an integral and increasingly vital component of software development, gaining even more prominence in emerging domains like Augmented Reality (AR) Applications. While AR technology has seen substantial integration with smartphones and other mobile devices, it still lacks standardized methodologies for software testing. This absence of standardized testing procedures poses significant challenges in the field, leading to limitations and potential inaccuracies during the testing process.

The unique nature of AR – blending digital elements with the real world – requires novel approaches to ensure software quality and reliability. The lack of established testing standards for AR applications complicates the assessment of usability, user experience, and overall functionality. Moreover, AR applications interact with diverse hardware and software ecosystems, further complicating the testing landscape.

This situation underscores the necessity for the development of comprehensive, standardized testing frameworks tailored to AR applications. Such frameworks would not only streamline the testing process but also enhance the accuracy and reliability of the results. As the field of AR continues to evolve and expand, the establishment of such standards will be crucial for advancing the quality and effectiveness of AR technologies.

In conclusion, the growing complexity and sophistication of AR applications demand a concerted effort toward the development of robust, standardized testing methodologies. This advancement will be critical in unlocking the full potential of AR technologies, ensuring their successful integration into various aspects of our lives and industries.

CHAPTER 2
SAPPROACH FOR TESTING AUGMENTED REALITY APPLICATIONS BY USING AUGMENTED REALITY APPLICATIONS

2.1. Theoretical Backgrounds

2.1.1. Software Testing and Quality Assurance Principles

The fundamental theories of software testing, including black-box testing, white-box testing, and automated testing, provide a foundation. These principles are adapted for AR environments, focusing on testing the unique aspects of AR applications such as spatial awareness, real-time interaction, and 3D rendering.

Early and Continuous Testing: Given the complexity of AR applications, which integrate real-time 3D rendering, user interaction, and often hardware components like cameras and sensors, early and continuous t That is, the sooner an error is detected, the less human and financial resources will be involved in its correctionesting is crucial to identify and resolve issues before they escalate. The cost of an error grows exponentially throughout the stages of the Software development lifecycle (SDLC). So, we must start looking for the bug when requirements are defined.

Requirement Traceability: This involves ensuring that the AR application meets specific requirements, such as accurate overlay of digital content onto the real world, responsive user interaction, and stable performance across various devices and environments.

Testing shows the presence of defects, not their absence: The purpose of testing is to identify and correct defects in software, but testing cannot ensure that the software is free of defects. If testing may not reveal any defects, that’s not proof that the software is flawless. Testing only reduces the probability of having undetected defects in the software that may affect its quality or functionality.

In AR, this principle underlines the importance of thorough testing, as defects can significantly disrupt the immersive experience.

Exhaustive testing is not possible: Exhaustive testing, which entails evaluating all possible combinations of inputs and preconditions, is unfeasible for QA teams due to its impracticality and cost. This process would require testing every conceivable module and scenario, posing a substantial challenge for any company.

Nevertheless, achieving high-quality software is attainable through meticulous planning and risk assessment. Focusing testing efforts on areas with potential software risks is the optimal approach to assure the software's quality.

Defect clustering: Defect clustering is a significant phenomenon in software testing, which aligns with the Pareto principle. According to this principle, roughly 80% of software issues can be traced back to only 20% of the modules. This phenomenon highlights the importance of focusing testing efforts on specific modules or features where the majority of defects tend to concentrate.

Factors contributing to defect clustering include the development of new features, frequent changes in existing modules, and dealing with legacy code. Testers and developers should be aware of this principle and prioritize testing in modules that have undergone frequent changes or have numerous dependencies. By doing so, they can efficiently identify and address defects, ensuring the delivery of a high-quality product to customers.

Identifying areas in AR applications that are prone to defects, such as complex user interactions or real-world integration points, allows more focused and effective testing efforts.

Testing is Context Dependent: Testing strategies and approaches vary depending on the context in which the software is developed and used. Different software applications, such as static websites, dynamic e-commerce sites, safety-critical industrial control software, or mobile e-commerce apps, require tailored testing methodologies to address their specific needs. For instance, safety considerations take precedence in aviation software, while user experience and speed are crucial for corporate websites.

Moreover, testing practices can differ between different stages of development, with Agile projects employing different methodologies than sequential lifecycle projects. Therefore, understanding the context in which software testing is conducted is essential for development and testing teams to design effective testing strategies.

Pesticide paradox: The concept of the Pesticide Paradox draws inspiration from the agricultural pesticide theory, where the repetitive use of pesticides leads to their ineffectiveness against pests over time. Similarly, in software testing, running the same test cases repeatedly can become less productive as they may not uncover new defects due to their redundancy. To address this paradox, it is essential to regularly review and update test cases, introducing new testing methods and techniques to detect previously undiscovered issues. This proactive approach ensures that testing remains effective and avoids falling into the trap of the Pesticide Paradox.

Absence of Errors Fallacy: It is a common fallacy to assume that a software product with minimal defects is ready for use. However, even if a software application is almost free of bugs, its true value lies in its ability to meet user requirements and solve business problems effectively. Simply focusing on error elimination is insufficient.

To ensure a software product's readiness, it is crucial to test it against both system requirements and user requirements. Testing alone cannot determine a product's readiness; user satisfaction and usability are equally important factors. If users find the software difficult to navigate or if it fails to address their needs, it can be considered a defect that jeopardizes the entire software product.

2.1.2. Software Testing and Quality Assurance Principles

Core AR theories, including the concepts of virtual overlays, user interaction in mixed reality environments, and spatial computing, are crucial. Understanding how AR elements interact with the real world and with the user is essential for designing tests that accurately assess an AR application's performance and usability.

- Virtual Overlays and Spatial Augmentation: This theory involves overlaying virtual objects onto the real world in a way that they appear to coexist in the same space. The challenge is to make these overlays as realistic and interactive as possible, taking into account the physical properties of the real environment.

- User Interaction in Mixed Reality: This concept focuses on how users interact with both real and virtual elements in an AR environment. It includes studying user interface design, interaction modalities (like gestures, voice commands, or touch), and user experience design specific to AR.
- Spatial Computing: This is a broad concept that refers to the ability of computers to interact with and understand the 3D space and objects within it. In AR, this involves processing and interpreting data about the physical environment, such as depth sensing, object recognition, and spatial mapping.

2.1.3. Cross-Application Communication and Interaction:les

Cross-application communication and interaction, especially in the context of Augmented Reality (AR), refers to the ability of different software applications or processes to communicate and interact with each other. It also includes data exchange formats and protocols that enable the testing tool to interact with and assess the tested application effectively. This concept is crucial in scenarios where multiple applications, possibly including AR applications, need to work in tandem or exchange data.

- **Inter-Process Communication (IPC)**: IPC is a fundamental concept where multiple processes (which can be parts of the same or different applications) exchange data. In the context of AR, this might involve an AR application communicating with a backend server application, or with other applications running on the same device.
- **APIs and Protocols**: Application Programming Interfaces (APIs) and communication protocols are essential for cross-application interaction. They define a set of rules and methods for applications to communicate. For AR applications, RESTful APIs, WebSocket, and other real-time communication protocols are commonly used.
- **Data Formats and Standards**: For effective communication, applications often need to agree on specific data formats and standards. In AR, this could include formats for 3D models, spatial data, and user interaction events.
- **Middleware and Frameworks**: Middleware and frameworks can facilitate cross-application communication by providing a layer of abstraction that handles the

communication details. This is particularly useful in complex AR systems that involve multiple components, such as tracking systems, content management systems, and user interfaces.

- **Synchronization**: When multiple applications interact, especially in real-time environments like AR, synchronization is crucial. This ensures that all interacting applications have a consistent and up-to-date view of the data and state of the system.
- **Networking and Connectivity**: For applications that are distributed over a network (e.g., cloud-based AR services), networking principles and connectivity issues become significant. This includes handling latency, bandwidth constraints, and connection stability.
- **Security and Privacy**: Secure communication channels are vital, especially when sensitive data is being transmitted. Encryption, authentication, and authorization mechanisms are key considerations in cross-application communication.
- **Scalability**: The communication and interaction mechanisms should be scalable to handle varying loads, which is important in AR applications that might need to support a large number of users or high volumes of data.
- **Error Handling and Robustness**: The system should be robust against communication failures or errors. This includes implementing retries, acknowledgments, and error-checking mechanisms.
- **User Context and Experience**: In AR, cross-application interaction should also consider the user context and experience. This includes how data exchange and application interaction impact the user's experience in an AR environment.

Cross-application communication and interaction in AR involve a combination of software engineering practices, networking principles, and user experience considerations. They ensure that multiple applications, including AR applications, can work together seamlessly, providing a coherent and integrated user experience.

2.1.4. Emulation and Simulation Theory

2.2. Proposed Approach

The methodology is based on the interaction of two AR programs, where data from one program must be transmitted to the other. The approach is such that both programs are independent, i.e., it does not imply the integration of an API to substitute incoming data for the system being tested. This, in turn, avoids adding potential vulnerabilities to the controlled program.

2.2.1. Methods and Challenges in Implementing AR Application Testing Interactions in android devices.

Before describing approaches to implement program interaction, it's important to note several complexities in such interactions. In the Android system, applications can only use one camera, and if at least one application uses the camera, others cannot use the device's camera. Therefore, it is not possible to set up program interaction through data transfer via a virtual camera on one device. Also, it is difficult to obtain virtual camera applications from official sources.

Moreover, even if it is possible to run two applications where one transmits data in the form of video from the camera to the other, there are issues with program operation services. A program in minimized mode can remain in working condition for a limited time, and the system begins to free up memory under high load. Since AR applications exert high load on mobile devices, closing the application for testing can be challenging to avoid.

Additionally, there is an issue with potential incompatibility of some programs with certain virtual machines and emulators, as well as the inability to install a virtual camera.

Before describing approaches, it's important to note that the tested application may have several types of information sources:

- **Broadcasting Video from the Screen**: This method involves testing in live mode, minimizing human error. However, it may suffer from delays in data transmission and reduced video quality.

- **Pre-recorded Video**: Using specially recorded videos transferred to a device for virtual camera transmission. This method is advantageous for automating testing and ensuring repeatability, though re-recording may be necessary in case of errors.
- **Photos and Screenshots**: This approach requires precise programming techniques and is suitable for testing that demands accurate imagery.

The first approach involves testing using cloud technologies. The most effective implementations are as follows:

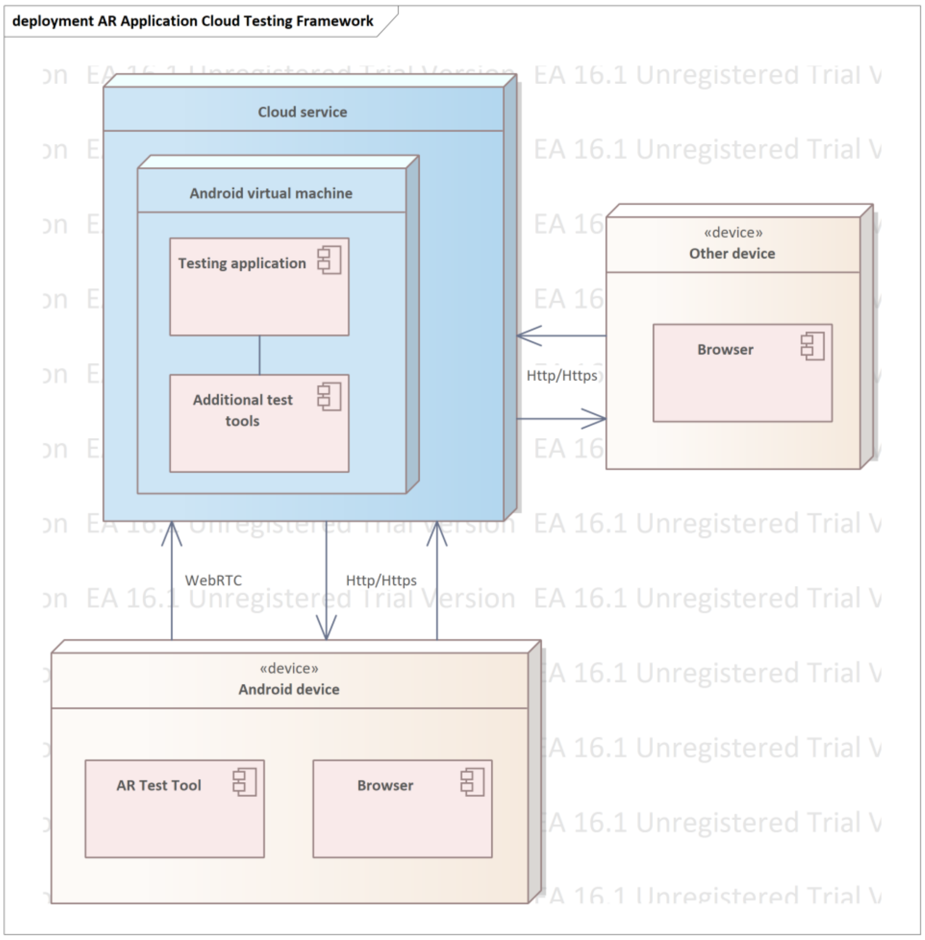

Fig. 2.1. Deployment Diagram. AR Application Cloud Testing Framework

a) The first implementation (see fig. 2.1.) is based on using the AR application for program testing on a mobile device, while the tested application is installed on an emulator set up in the cloud. During testing, data from the mobile device is transmitted to the emulator. This approach allows controlling the tested application from any location, including the device running the AR testing tool. The disadvantage of this approach is that when testing on the same device, the test program may be closed by the Android system.

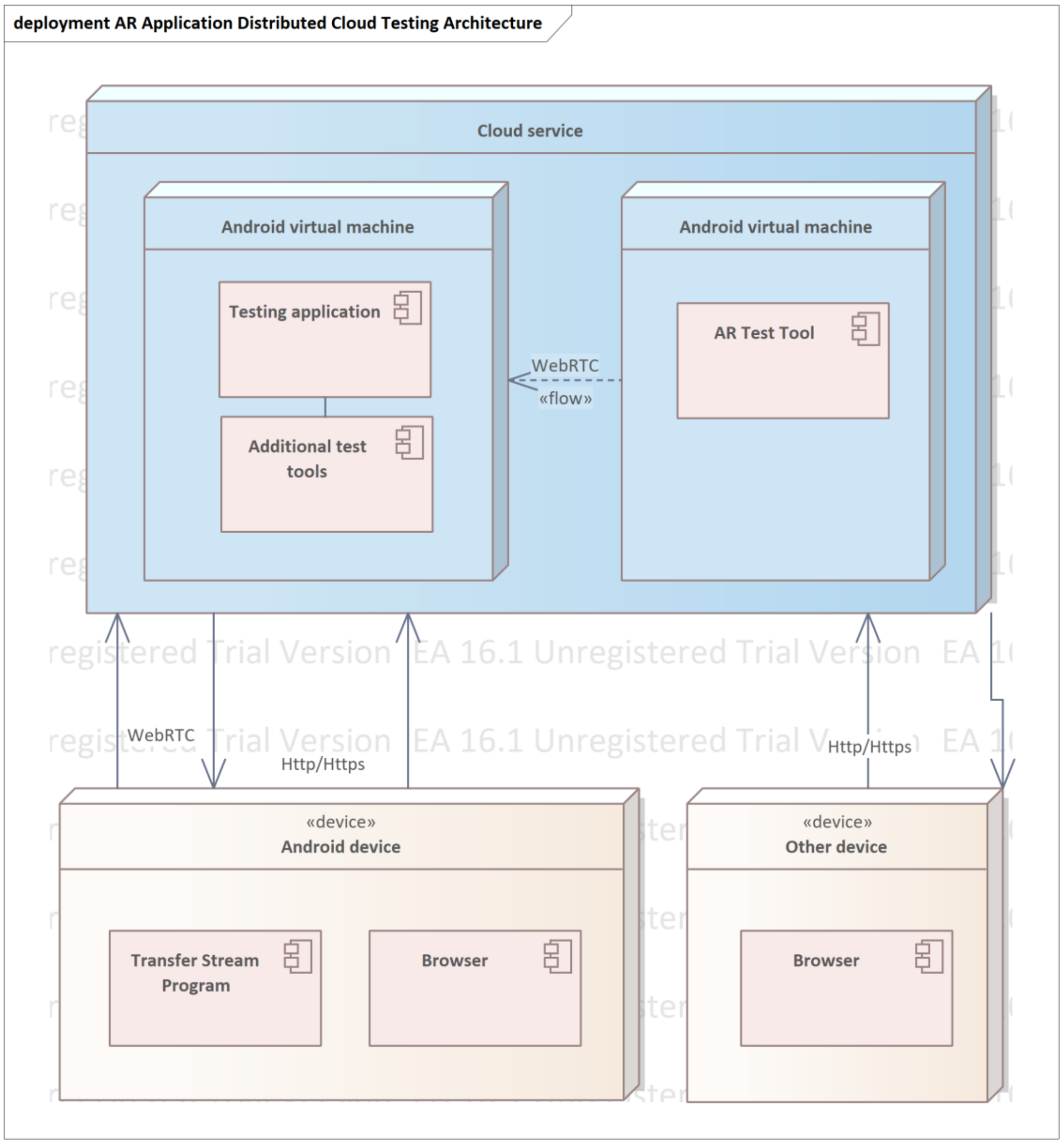

Fig. 2.2. Deployment Diagram. AR Application Cloud Testing Framework

b) The second implementation (see fig. 2.2.) involves placing both applications in separate cloud virtual machines. Data transmission occurs from the mobile device to the virtual machine with the AR testing tool, and then data is transferred from one virtual machine to another. This way, both programs can be controlled from any device without the risk of unexpected program closure. However, dependence on connectivity increases, as does the potential for noise and delays.

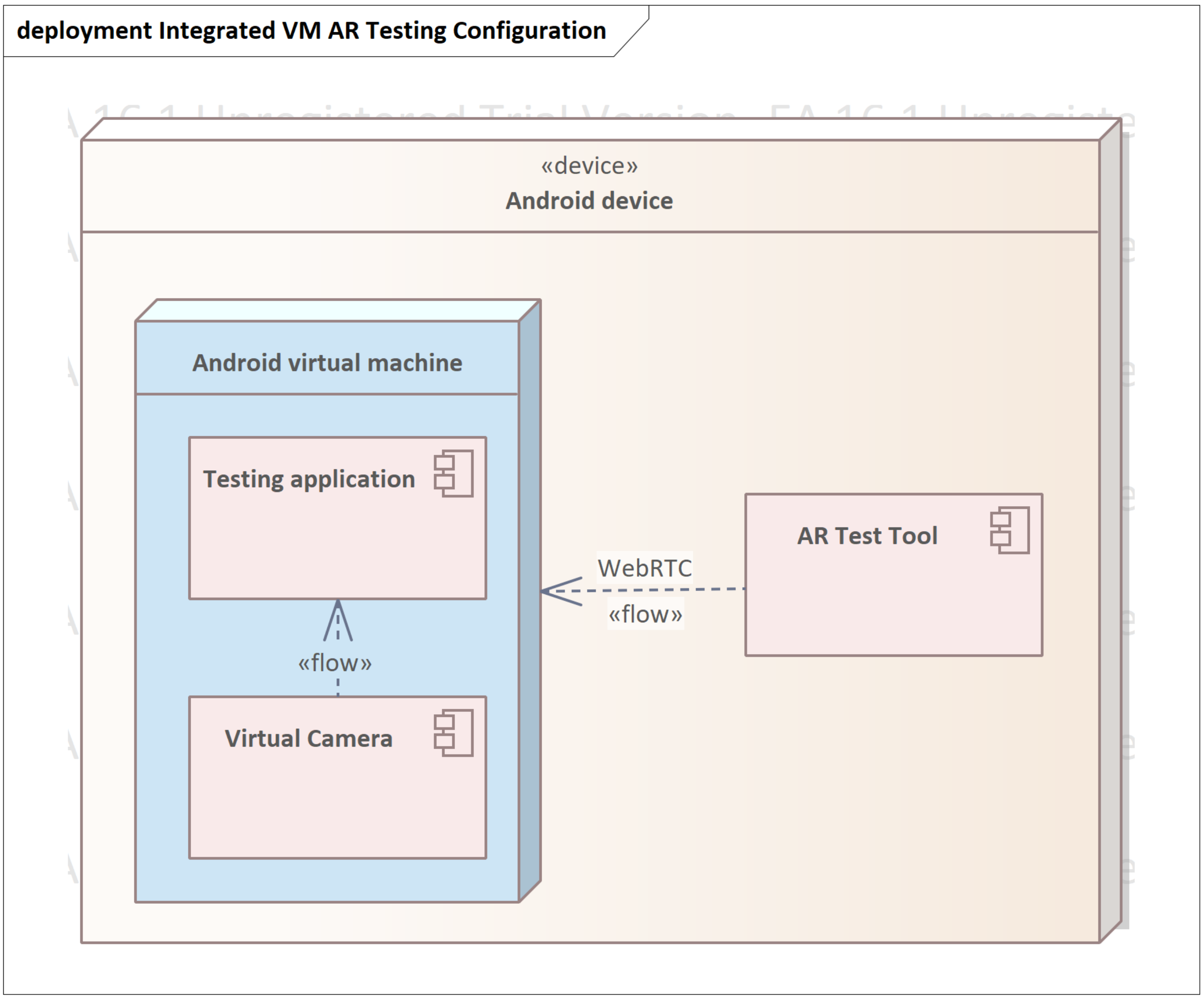

Fig. 2.3. Deployment Diagram. Integrated VM AR Testing Configuration

The second approach (see fig. 2.3.) involves using a virtual machine on the mobile device itself. In this case, the program for testing must itself perform the broadcast of its work, and the virtual machine must have the ability to set the source for the camera. This approach puts a tremendous load on the device itself but also does not depend on connectivity and works within a single device.

The main advantage of this approach is its autonomous nature. The convenience of using one device to perform one task - testing.

However, this approach does have significant implications in terms of device resource utilization. Running a virtual machine alongside the application places a high demand on the device's processing power, memory, and battery life. This could potentially lead to slower performance and might not accurately reflect the application's behavior in a typical usage scenario.

This method is best suited for preliminary testing stages where the focus is on functionality rather than performance. For performance and scalability testing, additional methods, possibly involving multiple devices or cloud-based solutions, would be more appropriate to get a comprehensive understanding of the application's behavior in real-world conditions.

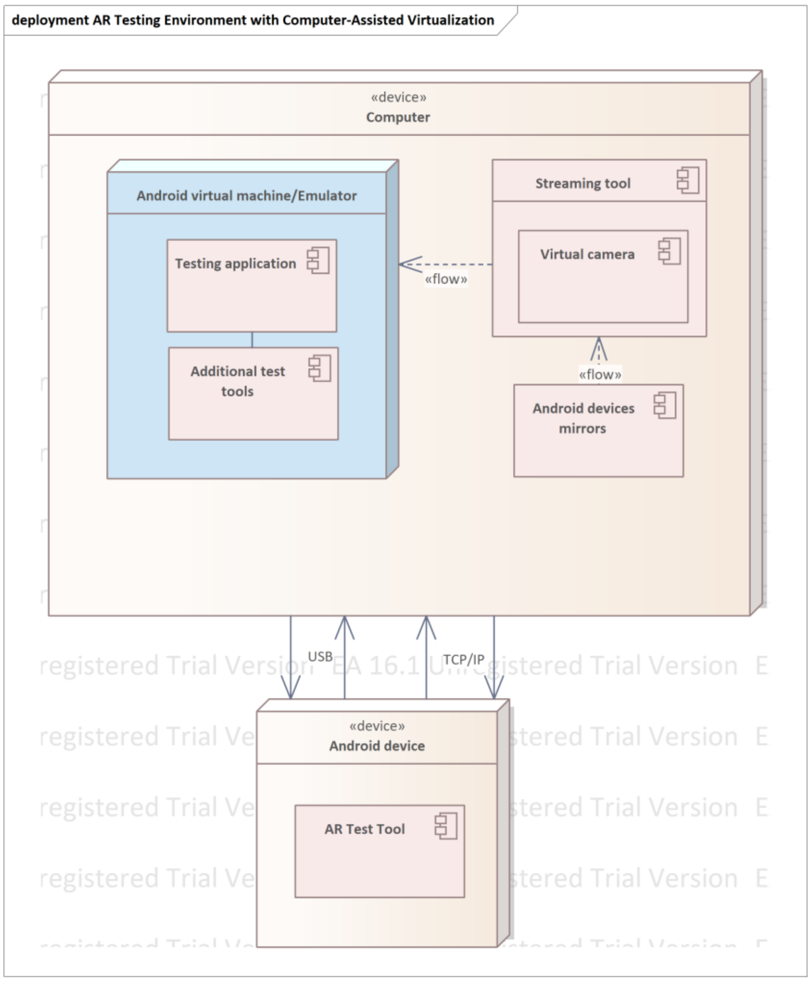

Fig. 2.4. Deployment Diagram. AR Testing Environment with Computer-Assisted Virtualization

The third approach (see fig. 2.4.) to AR application testing involves using a computer or laptop as an auxiliary device. In this setup, a virtual machine hosting the tested program is run on the computer. The key feature of this approach is the transmission of data from the AR device to the virtual machine, which can be facilitated either directly through the computer or via other connected devices.

This method leverages the computing power and resources of the computer to handle most of the testing workload. By offloading the processing and operational demands from the AR device to the computer, it allows for a more robust and stable testing environment. This can be particularly useful for applications that are resource-intensive or require a stable and controlled environment for accurate testing.

One of the benefits of this approach is the flexibility it offers in terms of testing configurations. Since the virtual machine is on a computer, it allows for easier manipulation and observation of the tested program's behavior. Additionally, it can facilitate the testing of different versions or configurations of the application without needing multiple physical devices.

However, this approach also requires a reliable connection between the AR device and the computer, whether it's via a local network, USB connection, or other means. The quality and reliability of this connection are crucial, as any interruption or lag could impact the testing process.

Moreover, setting up and configuring the virtual machine, along with ensuring the compatibility of the tested program with this environment, can add complexity to the testing process. It requires a certain level of technical expertise and understanding of both the AR technology and virtual machine management. Despite these challenges, this approach offers a versatile and powerful option for AR application testing, especially for developers and testers who have access to the necessary resources and technical skills.

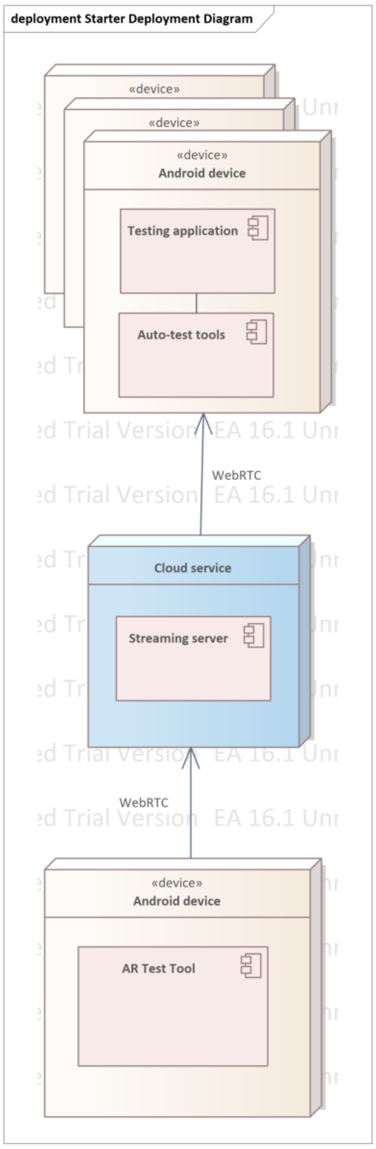

Fig. 2.5. Deployment Diagram. AR Streaming Deployment Architecture

The fourth approach involves using a single device with an application for testing and streaming the image to multiple devices. This approach, combined with the use of other tools, allows for automated testing of the application on multiple devices.

This method capitalizes on the concept of centralized testing and broadcasting. By streaming the test application's output from one device, it can be simultaneously observed and analyzed on multiple other devices. This setup is particularly beneficial for scenarios where the behavior of the application needs to be tested under different device conditions or operating environments.

One of the key advantages of this approach is the ability to conduct comprehensive testing without the need for multiple copies of the test application to be running on different devices. It simplifies the setup and reduces the resources needed for testing. Additionally, this approach can be integrated with automated testing tools to further streamline the testing process, allowing for more efficient identification of potential issues across different devices.

However, it's important to consider factors such as network reliability and bandwidth, as these can impact the effectiveness of streaming and, consequently, the testing process. The quality of the streaming should be sufficient to accurately represent the application's performance and any potential issues it may have. This method also requires a robust setup for capturing and streaming the application's output in real-time, which may involve additional technical complexities and resource requirements.

2.2.2. The sequence of implementation of the methodology

As no standards for development and testing are defined, certain sequences of steps may change and be supplemented depending on the evolution of the industry and the complexity and comprehensiveness of various projects. This overall process can be divided into the following subprocesses:

a) Define test objectives and scenarios: is the initial phase of AR application testing, where the primary goals, testing scenarios, usage variations, and functions subject to testing are established. Testing objectives are set to determine the ultimate outcomes of the testing process. Test scenarios outline specific situations, actions, and user interactions that will be simulated and verified within the AR application. The results of this phase play a central role in the testing process, ensuring that testing efforts align with project objectives and user expectations.

1) Study software requirements - involves a comprehensive analysis of both technical and user requirements. Technical requirements encompass the understanding of necessary hardware and software capabilities essential for augmented reality support, which include processing power, graphics, and sensor technologies. On the other hand, user requirements concentrate on the needs and preferences of end-users, emphasizing aspects such as usability, accessibility, and the integration of desired features or functionalities.

2) Examine product needs refers to a thorough analysis of both the functional and non-functional requirements of a testing app.

3) Research product audience involves the identification and understanding of the target users of the software. During this stage, testers gather information about the demographic data of the target users and their expectations to create profiles of potential software users, including factors such as age, gender, location, interests, technical proficiency, and any other relevant characteristics. This information can be used to form an understanding of user expectations, preferences, as well as information about possible scenarios for using the AR software. This information serves as the basis for adapting the testing process to ensure that the software meets the specific needs and desires of the target audience.

4) Prepare a List of Real-World Experiences is the phase of AR app testing, the
goal is to create a comprehensive list of real-world scenarios and experiences that the Application Under Test (AUT) should replicate. Testers identify and document specific user interactions, conditions, and situations users may encounter. These scenarios cover indoor and outdoor environments, different lighting conditions, various physical surfaces, and user actions. The aim is to ensure the AR app performs reliably across diverse real-world contexts, providing users with a seamless, immersive experience that meets their expectations.

5) Determine Supported Devices and Interactions: at this stage, specific devices
for testing and the types of user interactions supported by the tested program (AUT) are identified.

6) Define Test Coverage is focus is on outlining the specific areas and aspects of the application that will be subjected to testing. This includes identifying and defining the scope of testing, such as evaluating the user interface, functionality, and performance of the application. Test coverage ensures that all critical components and functionalities of the AR application are thoroughly examined and assessed during the testing process.

7) Confirm the Scope of Testing: During this phase, the testing team validates

and reiterates the defined scope of testing to ensure it aligns with the project's objectives. The scope encompasses the extent and boundaries of what will be tested, including the features, functionalities, and specific testing areas such as user interface, functionality, and performance. Confirming the scope of testing helps maintain focus and consistency throughout the testing process, ensuring that all critical aspects are appropriately covered.

b) Choose Testing Tools: involves the careful selection and configuration of the appropriate testing tools and resources. Testers identify and set up the necessary software, hardware, and frameworks to support the testing activities effectively. The choice of testing tools and resources is critical to ensure the thorough evaluation of the AR application's functionality, usability, and performance.

c) The preparation for testing using an AR test tool, as part of the methodology,
is distinct from the selection of the testing tool. This stage includes defining the method of establishing connectivity, configuring tools, and verifying the correct functioning of the test application. Depending on the chosen method of interaction, the steps of this stage may vary and can evolve over time. When considering testing using a virtual machine, the following stages can be identified:

1) The choice of an emulator or virtual machine involves selecting between these two software tools, each with its own operational characteristics. Depending on the collected data about the Android version and the required devices, either a particular virtualization tool may be chosen, or it may be necessary to abandon this approach altogether. This step should be the first when using a virtual machine approach, as proceeding with other steps in most cases can lead to significant time loss.

2) Verification of the correctness of sensor operation in a virtual machine or

emulator - virtual machines and emulators offer great possibilities for device configuration, but at the same time, this does not guarantee that the settings will be correct or that they will not require changes due to the specifics of their operation.

3) Check the operation of the program under test on an emulator/virtual machine - during this process, the tester needs to ensure that the program's core functions are working correctly. Particular attention should be paid to the camera launch, interface interaction, and the ability to use gestures. This step is necessary due to the limitations of emulators; for example, it is not possible to use gestures on a standard computer or laptop, and also due to the lack of support for some important libraries, the embeddedness of the Google Play service for AR, and limitations in the architecture and bit-depth of the processor.

4) Selection of tools and operating system for setting up system interactions. At this stage, depending on the chosen virtual machine or system, the tester needs to decide on the data transmission sources from the device with the AR testing tool to the virtual environment with the application under test. The choice of operating system also depends on the selected tools; for example, the Linux system offers higher performance for conducting tests and greater precision in settings, while the Windows system provides easier setup, and the Mac OS has better integration with virtual components, as with full-fledged devices.

5) Set Up Data Transfer. This stage involves configuring the interaction between the chosen devices with the help of defined tools. Initially, this step can be quite labor-intensive.

d) Set Up the Working Environment: During this phase, the testing team prepares the necessary devices and physical spaces required for conducting tests. The goal is to create a controlled testing environment that accurately simulates real-world conditions. This preparation ensures the effective execution of testing by replicating scenarios and contexts that users encounter when using the AR application.

e) Setting up the AR testing tool involves configuring and loading virtual objects, as well as determining their placement within the virtual environment. This stage focuses on preparing the digital elements and defining their positions on the virtual plane within the AR tool.

f) Define Testing Metrics: In this phase, specific metrics are outlined to evaluate the performance of the application. These metrics include factors such as latency, accuracy, and usability, among others. These metrics provide a structured and measurable way to assess the application's performance in key areas, helping to identify strengths and areas that require improvement.

g) Decide the Type of Testing: During this phase, the testing team determines the types of testing that will be conducted. This includes identifying whether functional testing, usability testing, performance testing, or other specific testing types are required. The decision on the type of testing to be conducted guides the testing strategy and ensures that the appropriate testing methods and criteria are applied to evaluate the AR application effectively.

h) Prepare Collaboration Tools: In this phase, the testing team sets up and configures the necessary tools and platforms to facilitate effective collaboration with developers. This includes implementing bug tracking systems, communication platforms, and other collaborative tools. The aim is to establish seamless communication and coordination between testers and developers to streamline issue reporting, resolution, and overall project collaboration.

i) Testing AR Application: This phase involves the actual testing of the AR application based on the prepared scenarios using the chosen tools and methodologies. The testing encompasses both automated and manual procedures to evaluate the application thoroughly. It also includes iterative testing and feedback loops, allowing for continuous improvement of the application. During this phase, detailed records are maintained to document testing processes, observations, and any issues discovered. Finally, a final evaluation is conducted to assess whether the AR application aligns with the initial objectives and requirements, ensuring that all criteria are met

Conclusion

This section has systematically presented and defined the foundational principles underlying the proposed methodology for testing AR applications. A crucial aspect that emerged is the current lack of standardized AR testing programs. While this absence allows for greater flexibility and freedom in developing testing methodologies, it simultaneously introduces a level of uncertainty. Addressing this uncertainty is pivotal in establishing robust and reliable testing practices for AR applications.

Furthermore, the section elaborated on a specific methodology that utilizes an AR testing tool. This tool plays a crucial role in the interaction of AR programs and facilitates the essential transfer of data between them. The methodology's effectiveness hinges on this interaction and data transfer capabilities, underscoring the need for innovative solutions in AR application testing.

In addition, a brief overview of potential interaction methods was provided, offering insights into various approaches for data transfer to the devices. These methods include cloud-based interactions, the use of virtual machines, and leveraging external computing devices, each with its unique advantages and challenges.

Lastly, a comprehensive description of the methodology and the steps for its implementation was outlined. This detailed account serves as a guide for effectively employing the methodology in practical testing scenarios.

In conclusion, the development and refinement of this methodology represent a significant contribution to the field of AR application testing. As AR technology continues to evolve, the adaptation and enhancement of these testing approaches will be crucial for ensuring the reliability and effectiveness of AR applications in various domains.

CHAPTER 3
DESCRIPTION OF AR TESTING TOOLS

3.1. Software Product Specification

The software product is a tool that uses AR technology for testing other AR applications.

The mission of the software is to assist in the development of the educational environment and its components, including the participants of this environment. More specifically, its mission is to provide an information base of competencies, which includes the analysis and collection of modern requirements for specialists in certain fields of activity, as well as the competencies already present in the students of the department to assist in forming a system of professional and positional adaptation for graduates.

The mission of the software is also to assist in conducting works related to the testing of AR applications. More specifically, its mission is to form a surrounding environment that will contain the necessary real and virtual objects for checking the functionality of the program, as well as its behavior and functioning in a mixed reality environment.

According to the need to form an environment and place virtual applications, the software tool must be related to Marker-Less AR.

Accordingly, the following functional capabilities of the software application can be formed:

- Recognition of surfaces for placing virtual objects
- Placement of virtual objects
- Deletion of objects
- Changing the position of an object in the environment
- Changing the size of virtual objects
- Changing the appearance of objects
- Forming templates
- Selecting virtual objects for placement

3.2. Tools that were used in the development of the application.

The development of the software product was decided to be conducted on the cross-platform development environment Unity, using the integrated development environment (IDE) Visual Studio and the text editor VS Code.

The programming language chosen for writing the program was C# using the .NET Standard 2.1 specification and the Mono framework. This combination offers a robust platform for developing versatile and high-performance software.

The development of graphical 2D elements was carried out using the GNU Image Manipulation Program (GIMP).

Version control, an essential aspect of software development, was managed using the Git system, ensuring efficient tracking and management of code changes.

3.2.1. Tools description

Unity is a development platform widely used for creating interactive media such as video games, architectural visualizations, and real-time 3D animations. It's known for its versatility and ease of use, enabling developers to deploy projects across various platforms including PCs, consoles, mobile devices, and VR/AR systems.

Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services, and mobile apps. Visual Studio supports a range of programming languages, including C#, VB.NET, C++, and F#, and features tools for developing and debugging code, managing source code repositories, and deploying applications.

C# (pronounced "C Sharp") is a modern, object-oriented programming with a safe typing system for the .NET platform.

Mono is an open-source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime (CLR). It was originally developed by Ximian, which was later acquired by Novell, and is currently maintained by the .NET Foundation and the Mono community.

.NET Standard 2.1 is a formal specification of .NET APIs that are intended to be available on all .NET implementations. This standard facilitates the development of libraries that are compatible across multiple .NET platforms, enabling developers to write code that can run on various systems without modification.

Git is a distributed version control system, widely used for tracking changes in source code during software development. It is designed for speed, data integrity, and support for distributed, non-linear workflows.

GIMP is a free and open-source raster graphics editor used for image retouching and editing, free-form drawing, converting between different image formats, and more specialized tasks. GIMP is available for various operating systems

3.2.2. Libraries and technologies

Package Manager is a tool of Unity Editor that facilitates the discovery, installation, and management of Unity packages. Unity packages are collections of assets, tools, and plugins that can be used to add functionality and content to Unity projects.

AR Foundation is a framework developed by Unity Technologies for building augmented reality (AR) experiences. It provides a common API that works across both Android and iOS devices, enabling developers to create AR applications that are deployable on multiple platforms without having to write platform-specific code.

Raycasting is a computational technique used in computer graphics and simulation to simulate the behavior of rays or lines as they interact with objects in a 2D or 3D environment. It is commonly used in various applications such as 3D computer graphics, virtual reality, and game development for tasks like collision detection, rendering, and visibility determination.

The **Google ARCore XR Plugin** is a component designed for integrates Google's ARCore technology into Unity, enabling developers to create augmented reality (AR) experiences for Android devices. ARCore is Google's platform for building AR applications. It uses the phone's camera to understand and interact with the world.

OpenXR is an open and royalty-free standard for creating and deploying virtual reality (VR) and augmented reality (AR) applications and devices. It is designed to provide a unified and standardized interface for different VR and AR platforms, allowing developers to write their applications once and have them work seamlessly across various hardware and software ecosystems.

ProBuilder is a plugin for the Unity game engine that allows developers and 3D artists to easily create, edit, and prototype 3D models directly within the Unity editor. It is a powerful and versatile tool that streamlines the 3D modeling and level design process.

TextMeshPro is an advanced text rendering and layout system for the Unity game engine. It is designed to provide enhanced text rendering and formatting capabilities compared to Unity's built-in Text component. TextMeshPro is especially useful for creating visually appealing and high-quality text in interactive applications.

The **Universal Render Pipeline** (formerly known as the Lightweight Render Pipeline or LWRP) is a rendering system that is designed to provide high-quality graphics and performance while remaining efficient and lightweight. It is a versatile rendering pipeline suitable for a wide range of platforms and devices.

The **XR Interaction Toolkit** is a set of tools and features provided by Unity to facilitate the development of XR (Extended Reality) applications, including virtual reality (VR) and augmented reality (AR) experiences. This toolkit simplifies the process of creating immersive and interactive environments in Unity, allowing developers to focus more on the unique aspects of their applications rather than the foundational elements of XR development.

XR Plugin Management is tool that allows developers to manage and configure various XR (Extended Reality) platforms and technologies for building Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR) applications. XR Plugin Management provides a unified interface for handling different XR platforms, making it easier to develop cross-platform XR applications.

OpenJDK (Open Java Development Kit) is an open-source implementation of the Java Platform, Standard Edition (Java SE). It provides a free and open-source alternative to Oracle's Java Development Kit (JDK), which is the official reference implementation of Java SE. OpenJDK is maintained and developed by the open-source community and is widely used for Java application development. OpenJDK in Unity is used to complement the Android SDK.

The **Android SDK** (Software Development Kit) is a set of development tools provided by Google to create applications for the Android platform. The SDK includes a comprehensive set of development tools, including libraries, a debugger, a handset emulator, documentation, sample code, and tutorials.

The Android Native Development Kit (NDK) is a toolset that allows developers to implement parts of their app using native-code languages such as C and C++. It is used when performance is critical for the app, such as for computationally intensive applications like game engines.

The **"Native Gallery for Android & iOS"** is a Unity asset designed to enhance the interaction with the device's gallery or photo library on both Android and iOS platforms.

LiteDB is an open-source NoSQL database that is lightweight and designed for use in .NET applications. It is serverless and fully embedded, meaning it doesn't require installation of an additional database server, but rather it runs directly within the application. LiteDB stores data in a single file using a document-oriented approach, similar to how MongoDB operates.

3.3. Application class structure

During the software development process, 17 classes and 3 interfaces were identified. The overall structure of these classes can be seen in the class diagrams (see fig. 3.1 – 3.2).

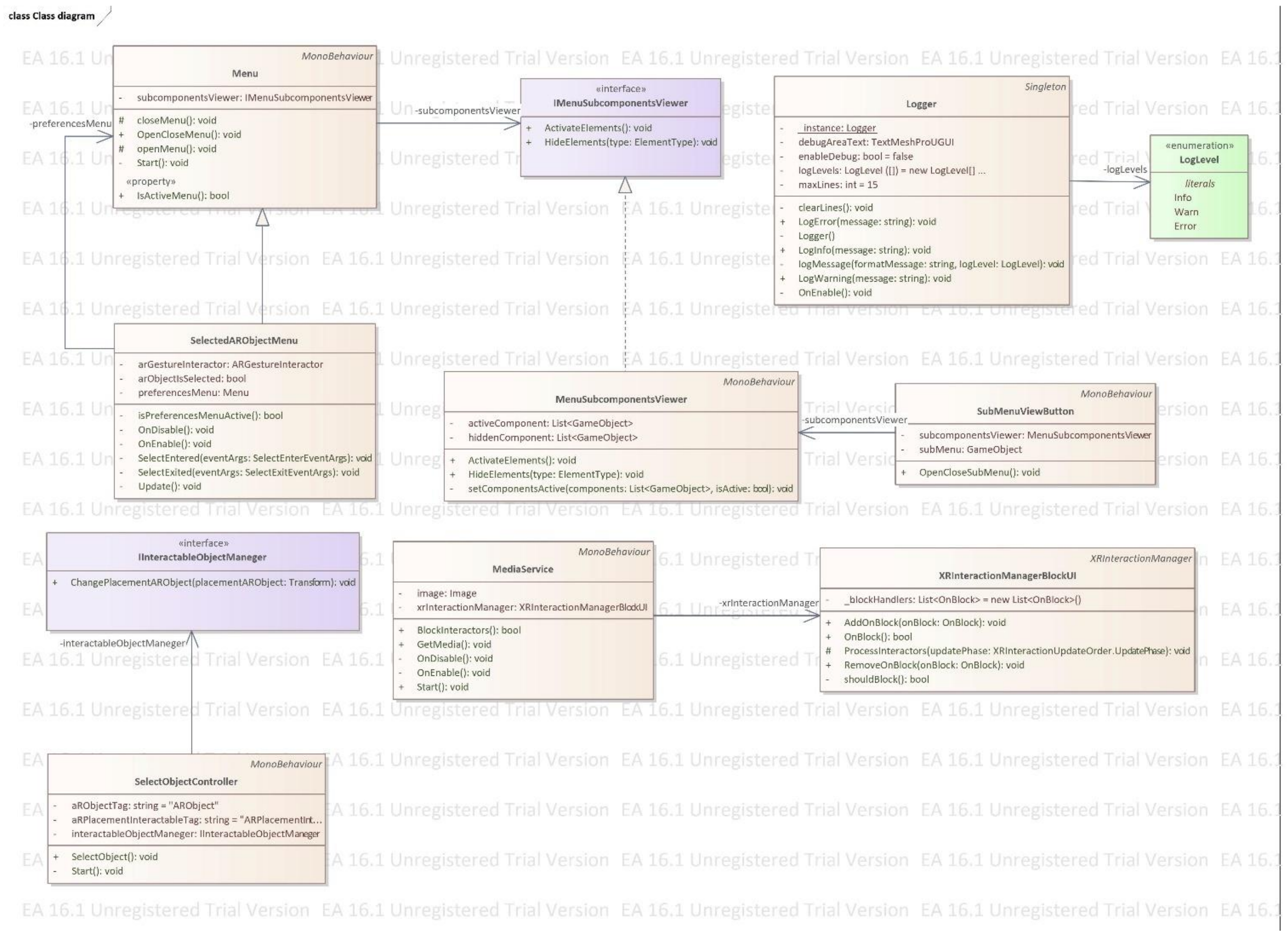
class Class diagram
MonoBehaviour
Menu
- subcomponentsViewer: IMenuSubcomponentsViewer
# closeMenu(): void
+ OpenCloseMenu(): void
# openMenu(): void
- Start(): void
«property»
+ IsActiveMenu(): bool
-preferencesMenu
-subcomponentsViewer
«interface»
IMenuSubcomponentsViewer
+ ActivateElements(): void
+ HideElements(type: ElementType): void
Singleton
Logger
- _instance: Logger
- debugAreaText: TextMeshProUGUI
- enableDebug: bool = false
- logLevels: LogLevel ([]) = new LogLevel[] ...
- maxLines: int = 15
- clearLines(): void
+ LogError(message: string): void
- Logger()
+ LogInfo(message: string): void
- logMessage(formatMessage: string, logLevel: LogLevel): void
+ LogWarning(message: string): void
- OnEnable(): void
-logLevels
«enumeration»
LogLevel
literals
Info
Warn
Error
SelectedARObjectMenu
- arGestureInteractor: ARGestureInteractor
- arObjectIsSelected: bool
- preferencesMenu: Menu
- isPreferencesMenuActive(): bool
- OnDisable(): void
- OnEnable(): void
- SelectEntered(eventArgs: SelectEnterEventArgs): void
- SelectExited(eventArgs: SelectExitEventArgs): void
- Update(): void
MonoBehaviour
MenuSubcomponentsViewer
- activeComponent: List<GameObject>
- hiddenComponent: List<GameObject>
+ ActivateElements(): void
+ HideElements(type: ElementType): void
- setComponentsActive(components: List<GameObject>, isActive: bool): void
-subcomponentsViewer
MonoBehaviour
SubMenuViewButton
- subcomponentsViewer: MenuSubcomponentsViewer
- subMenu: GameObject
+ OpenCloseSubMenu(): void
«interface»
IInteractableObjectManeger
+ ChangePlacementARObject(placementARObject: Transform): void
-interactableObjectManeger
MonoBehaviour
MediaService
- image: Image
- xrInteractionManager: XRInteractionManagerBlockUI
+ BlockInteractors(): bool
+ GetMedia(): void
- OnDisable(): void
- OnEnable(): void
+ Start(): void
-xrInteractionManager
XRInteractionManager
XRInteractionManagerBlockUI
- _blockHandlers: List<OnBlock> = new List<OnBlock>()
+ AddOnBlock(onBlock: OnBlock): void
+ OnBlock(): bool
# ProcessInteractors(updatePhase: XRInteractionUpdateOrder.UpdatePhase): void
+ RemoveOnBlock(onBlock: OnBlock): void
- shouldBlock(): bool
MonoBehaviour
SelectObjectController
- aRObjectTag: string = "ARObject"
- aRPlacementInteractableTag: string = "ARPlacementInt...
- interactableObjectManeger: IInteractableObjectManeger
+ SelectObject(): void
- Start(): void

Fig 3.1. Class diagram (part 1)

Menu - A class responsible for managing menu components. IMenuSubcomponentsViewer - An interface that defines functionality for controlling the display of menu items and also differentiates components into active - constantly active components and hidden - components that are hidden by default.

MenuSubcomponentsViewer - Implements the IMenuSubcomponentsViewer interface and defines the behavior for displaying and hiding components when opening and closing the menu.

SubMenuViewButton - A logic class for controlling the display of submenus. Determines the submenu items that need to be displayed and the logic for closing other submenus. Logger - A class for logging information and displaying logs on devices. Designed to track logs during program operation. Also inherits from the class Singleton<T> and implements the singleton pattern.

Singleton<T> - A generic class for implementing a single logic for creating a class according to the singleton pattern.

SelectedARObjectMenu - A class for managing the display of a menu for a selected AR object. Tracks the selection operation of an AR object and displays menu components upon selection. Hides components while another menu is open until it is closed.

IInteractableObjectManeger - An interface that defines the logic for replacing the template in an interactable object - determines how the object will appear.

SelectObjectController - A class that tracks the selection of a template for object replacement.

MediaService - A submenu component that blocks the use of AR tools during the setting of images on virtual objects. The class also uses native logic to enable image upload to the software tool.

XRInteractionManagerBlockUI - An extended class of the AR tools manager, which additionally implements the function of blocking the operation of AR tools when interacting with UI components, as well as the ability to configure blocking under certain conditions.

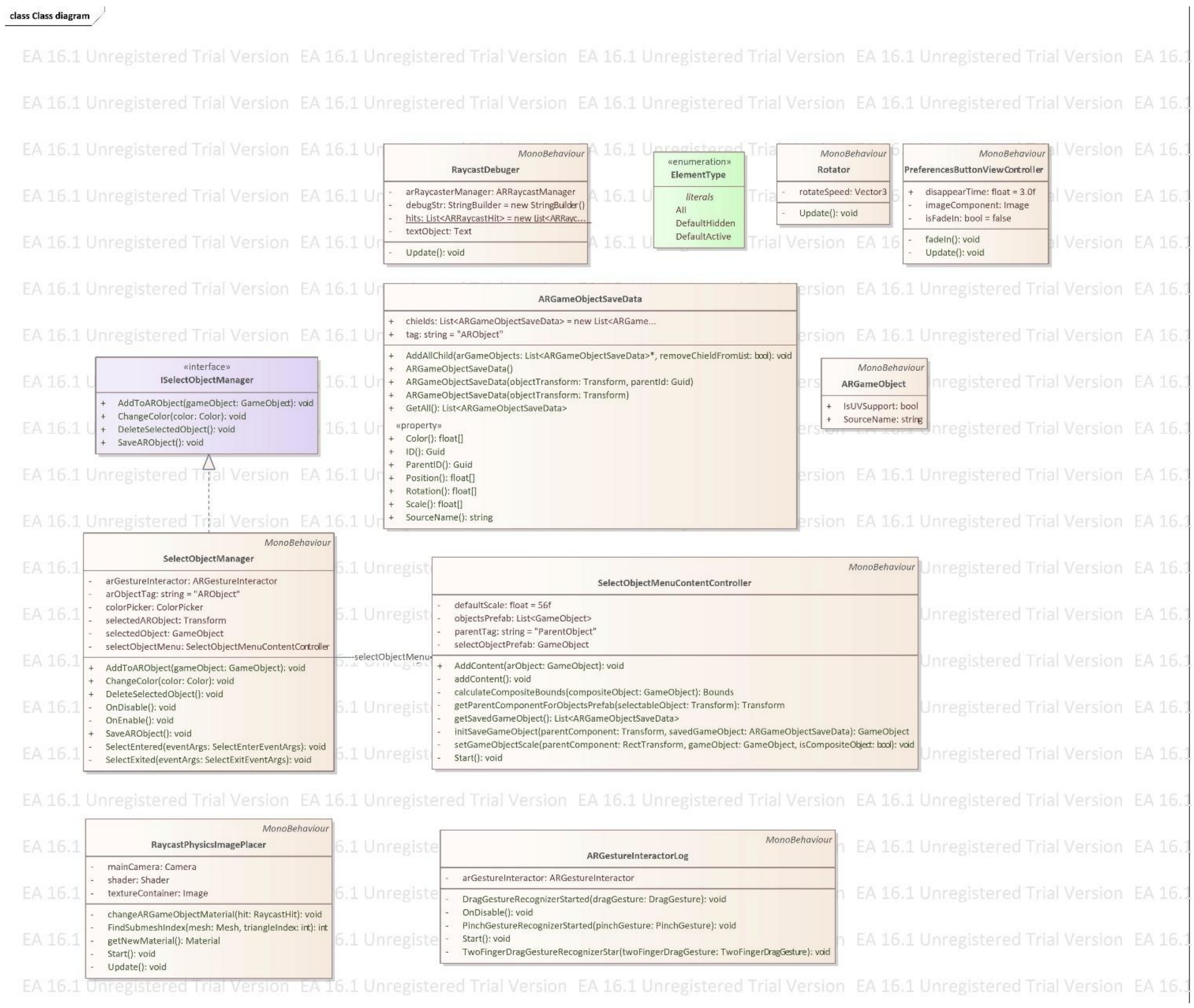
class Class diagram
MonoBehaviour
RaycastDebuger
- arRaycasterManager: ARRaycastManager
- debugStr: StringBuilder = new StringBuilder()
- hits: List<ARRaycastHit> = new List<ARRayc...
- textObject: Text
- Update(): void
«enumeration»
ElementType
literals
All
DefaultHidden
DefaultActive
MonoBehaviour
Rotator
- rotateSpeed: Vector3
- Update(): void
MonoBehaviour
PreferencesButtonViewController
+ disappearTime: float = 3.0f
- imageComponent: Image
- isFadeIn: bool = false
- fadeIn(): void
- Update(): void
ARGameObjectSaveData
+ chields: List<ARGameObjectSaveData> = new List<ARGame...
+ tag: string = "ARObject"
+ AddAllChild(arGameObjects: List<ARGameObjectSaveData>*, removeChieldFromList: bool): void
+ ARGameObjectSaveData()
+ ARGameObjectSaveData(objectTransform: Transform, parentId: Guid)
+ ARGameObjectSaveData(objectTransform: Transform)
+ GetAll(): List<ARGameObjectSaveData>
«property»
+ Color(): float[]
+ ID(): Guid
+ ParentID(): Guid
+ Position(): float[]
+ Rotation(): float[]
+ Scale(): float[]
+ SourceName(): string
«interface»
ISelectObjectManager
+ AddToARObject(gameObject: GameObject): void
+ ChangeColor(color: Color): void
+ DeleteSelectedObject(): void
+ SaveARObject(): void
MonoBehaviour
ARGameObject
+ IsUVSupport: bool
+ SourceName: string
MonoBehaviour
SelectObjectManager
- arGestureInteractor: ARGestureInteractor
- arObjectTag: string = "ARObject"
- colorPicker: ColorPicker
- selectedARObject: Transform
- selectedObject: GameObject
- selectObjectMenu: SelectObjectMenuContentController
+ AddToARObject(gameObject: GameObject): void
+ ChangeColor(color: Color): void
+ DeleteSelectedObject(): void
- OnDisable(): void
- OnEnable(): void
+ SaveARObject(): void
- SelectEntered(eventArgs: SelectEnterEventArgs): void
- SelectExited(eventArgs: SelectExitEventArgs): void
-selectObjectMenu
MonoBehaviour
SelectObjectMenuContentController
- defaultScale: float = 56f
- objectsPrefab: List<GameObject>
- parentTag: string = "ParentObject"
- selectObjectPrefab: GameObject
+ AddContent(arObject: GameObject): void
- addContent(): void
- calculateCompositeBounds(compositeObject: GameObject): Bounds
- getParentComponentForObjectsPrefab(selectableObject: Transform): Transform
- getSavedGameObject(): List<ARGameObjectSaveData>
- initSaveGameObject(parentComponent: Transform, savedGameObject: ARGameObjectSaveData): GameObject
- setGameObjectScale(parentComponent: RectTransform, gameObject: GameObject, isCompositeObject: bool): void
- Start(): void
MonoBehaviour
RaycastPhysicsImagePlacer
- mainCamera: Camera
- shader: Shader
- textureContainer: Image
- changeARGameObjectMaterial(hit: RaycastHit): void
- FindSubmeshIndex(mesh: Mesh, triangleIndex: int): int
- getNewMaterial(): Material
- Start(): void
- Update(): void
MonoBehaviour
ARGestureInteractorLog
- arGestureInteractor: ARGestureInteractor
- DragGestureRecognizerStarted(dragGesture: DragGesture): void
- OnDisable(): void
- PinchGestureRecognizerStarted(pinchGesture: PinchGesture): void
- Start(): void
- TwoFingerDragGestureRecognizerStar(twoFingerDragGesture: TwoFingerDragGesture): void

Fig 3.2. Class diagram (part 2)

RaycastDebugger – A class for logging the operation of raycast technology, tracking elements interacted with by emitted rays.

Rotator – A class that allows the addition of rotation functionality to a linked component, enabling it to rotate around a specified axis.

PreferencesButtonViewController – A class that defines the logic for time-based fading of menu buttons.

ARGameObjectSaveData – A class for saving active interactive objects in a database, as well as creating templates based on them.

ARGameObject – Responsible for the properties of AR objects.

ISelectObjectManager – An interface that defines methods for working with a selected interactive object.

SelectObjectManager – Inherits from ISelectObjectManager and implements methods for managing the state of an interactive object.

SelectObjectMenuContentController – A class that manages templates, as well as being responsible for creating UI components for template selection.

RaycastPhysicsImagePlacer – Implements the invocation of physical raycasting for interacting with virtual 3D objects and applying selected images onto them.

ARGestureInteractorLog – A class that subscribes to events of AR tools for further logging of their operations.

3.4. Use cases of using AR application tool

Options for using the application are presented in Figure 3.3

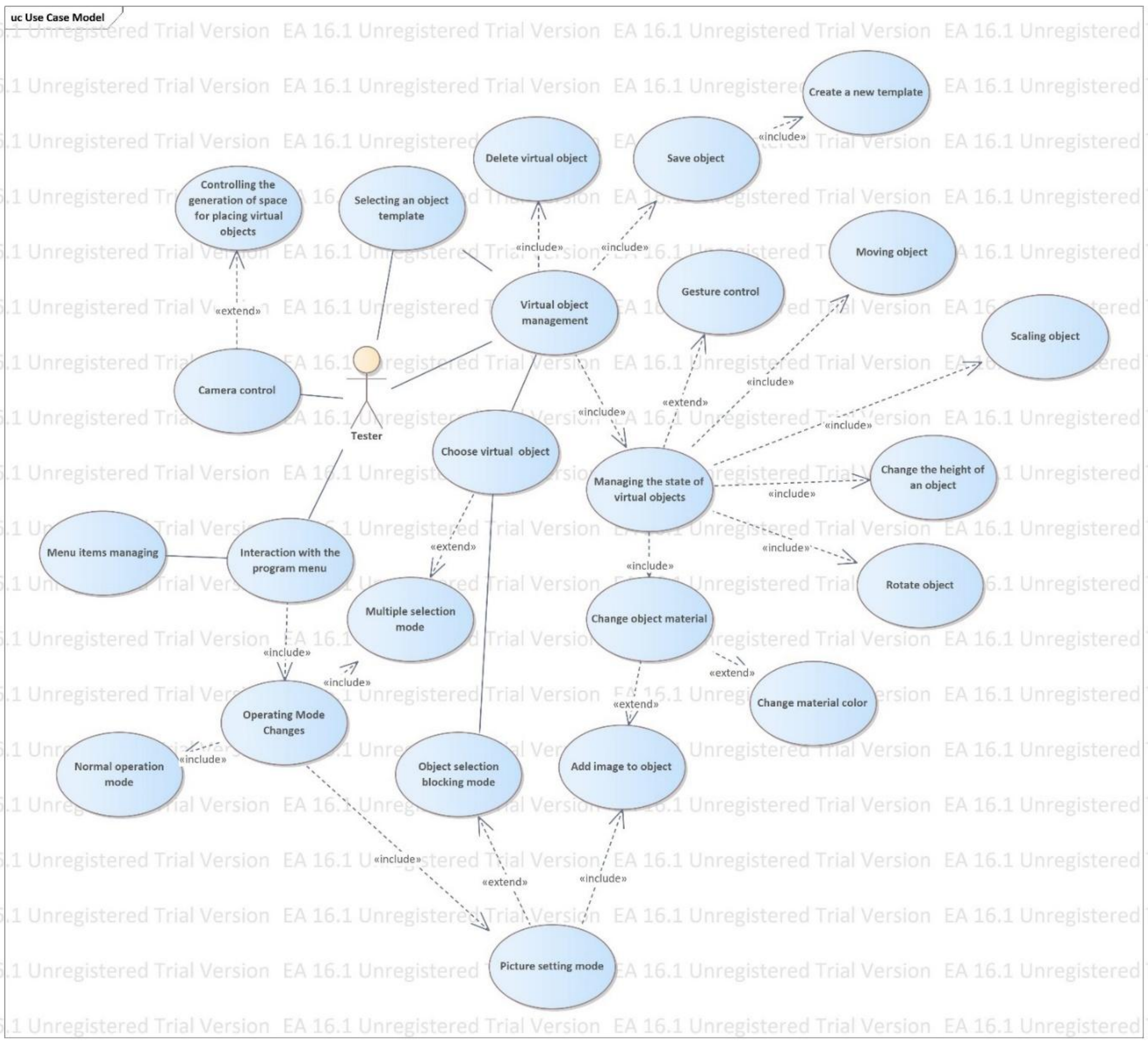

Fig 3.2. Use case diagram for tester

Let's take a closer look at the diagram of the tester's choice options.

Interaction with the program menu - enables the tester to use UI elements of the interface without interacting with the virtual AR environment and its components. This includes the ability to change operating modes.

Operating mode changes - the tester's ability to interact with the application interface, which changes the program's behavior. This includes normal operation mode, picture setting mode, and multiple selection mode.

Normal operation mode - the tester's ability to work in the standard mode.

Picture setting mode - the ability for the tester to work in image selection and installation mode.

Add image to object - the ability of the tester to set images for an object or its face, depending on the settings of the virtual object.

Menu items visible management - using UI elements, the user can control the visibility of some system components.

Object selection blocking mode - the user's ability to interact with the AR scene using rays for more precise targeting of virtual objects for further adding to them or their edges of selected images.

Multiple selection mode – the ability of the tester to select multiple objects for simultaneous interaction with them or for unification. This mode does not work simultaneously with other modes, and the combination is performed with the first selected component.

Camera control - the ability of the tester to control the position of his device, thereby changing the position of the camera.

Controlling the generation of space for placing virtual objects - the recognition and generation of planes for placing virtual objects depend on the control of the user’s device.

Management of virtual objects - the tester’s ability to manage virtual objects. This includes features such as deleting a virtual object, saving an object, selecting a virtual object, and managing the state of virtual objects.

Selection of a virtual object - the tester’s ability to select a virtual object for further manipulations.

Delete virtual object - allows the tester to delete the selected object.

Save object - allows the tester to save the selected object. This includes the ability to create a template.

Create a new template - the tester can create a new template based on a saved object.

Selecting an object template - the ability to select an object template that will be used to create a virtual object when placing it on a surface.

Managing the state of virtual objects - the ability of the tester to set the position in space, as well as change the virtual object. This includes features such as gesture control, moving an object, scaling an object, changing the height of an object, rotating an object, and changing object material.

Gesture control - the tester’s ability to control a virtual object using gestures on the smartphone screen.

Moving object - the ability of the tester to move a virtual object within the territory area for placing virtual objects.

Object scaling - the tester’s ability to change the scale of the selected virtual object.

Change the height of an object - the ability to change the height of virtual objects.

Rotate object - the ability to change the angle of placement of a virtual object.

Change object material - the tester can change the material of an object. This includes changing the material color.

Change material color - allows the tester to set the material color for the selected virtual object. Experience: Users expect a stable and smooth AR experience.

Conclusion

This section has systematically outlined the functional requirements for a tool designed to test advanced reliability programs. These requirements form the backbone of the tool's development and ensure that it meets the specific needs of reliability testing in complex software environments.

In addition, it was presented a comprehensive list of the main technologies, frameworks, and tools utilized in the development of the testing program was presented. This list provides insights into the technical stack and the rationale behind the selection of each component, reflecting the latest trends and best practices in software development.

Also delineated the primary capabilities available to the user of the tester program. This aspect is crucial as it directly impacts the user experience and the effectiveness of the program in conducting thorough and efficient reliability tests.

Visual representation of the program structure was done using a class diagram, showcasing the relationships and interactions between different classes. This diagram serves as a valuable tool for understanding the program’s architecture and for guiding future modifications or enhancements.

CHAPTER 4
APPLICATION OF THE PROPOSED APPROACH

4.1. Definition of the object of testing

An application called Zappar was chosen for testing. Zappar is an augmented reality (AR) application that allows users to create and interact with an AR world. It uses marker-based technology to trigger AR content. When users scan a Zappar code or a physical object designated as a token (such as product packaging, advertising, or even clothing), the app overlays the digital content onto the real world viewed through the device's camera.

This program is part of a large infrastructure project that includes work for both beginners and professionals, so the digital content can vary from simple animations and videos to interactive games and 3D models. Zappar is often used in marketing and advertising to create engaging and attractive brands, but it also has applications in education and entertainment.

The app is designed to be user-friendly, allowing you to not only consume AR content, but also create your own AR experience. This makes it a popular choice for companies looking to incorporate AR into their marketing strategies, as well as educators and creators who want to explore the potential of AR technology.

4.2. Define test objectives and scenarios:

According to the specific requirements of the software, the following technical requirements can be identified: compatibility with various models and types of mobile devices. There should also be compatibility between different operating systems, ensuring productive performance in poor environmental conditions, supporting various types of graphics, and optimal resource utilization – as the program interacts with an additional source, stable data exchange control is necessary. Additionally, the ability to support multiple interactions simultaneously is required.

As for user requirements, the interface should provide intuitiveness and simplicity in navigation. Users should also be provided with sufficient information and be able to interact with the AR content.

4.2.1. Examine product needs.

Functional Requirements:

- Marker Detection and Tracking: The app must effectively detect and track markers in various environments to trigger AR experiences.
- Content Rendering: Ability to render 3D models, animations, videos, and interactive content smoothly in an AR setting.
- User Interaction: Support for user interactions with AR content, such as touch gestures, motion tracking, or voice commands.
- Content Management: Features for managing AR content, including downloading, updating, and caching.
- Integration with Other Services: If applicable, integration with external services like social media, cloud storage, or analytics.

- Marker Detection and Tracking: The app must effectively detect and track markers in various environments to trigger AR experiences.
- Content Rendering: Ability to render 3D models, animations, videos, and interactive content smoothly in an AR setting.
- User Interaction: Support for user interactions with AR content, such as touch gestures, motion tracking, or voice commands.
- Content Management: Features for managing AR content, including downloading, updating, and caching.
- Integration with Other Services: If applicable, integration with external services like social media, cloud storage, or analytics.

- Cross-Platform Support: Ensuring compatibility and optimized performance across different devices and operating systems.

Non-Functional Requirements:

- Performance: The app should function smoothly without significant lags or crashes, even when rendering complex AR scenes.
- Usability: User-friendly interface, intuitive navigation, and ease of use for a wide range of users.
- Scalability: Ability to handle an increasing amount of work and number of users without performance degradation.
- Reliability: Consistent performance over time, with minimal downtime or errors.
- Compatibility with another device
- Network Efficiency: Optimized data usage, especially important for mobile users with limited data plans.

4.2.2. The audience of the software product

Due to the specific infrastructure that the Zappar product is part of, it has a broad user audience, ranging from children and their parents who use it for entertainment and learning, to 3D artists who use it to review their own work. Several user groups can be identified:

- Children and Their Parents or Guardians: They use the app for entertainment and educational purposes.
- General Users: These users engage with the software for entertainment purposes.
- Advertisers: They utilize the app for presenting unique interactive advertising.
- Artists and Designers: Use the software for skill development and artistic growth.

4.2.3. Real-world scenarios

Given the described user groups, it's challenging to specify statistical conditions in which the software application can operate, thus potential scenarios include:

Indoor Environments

- Home Settings: Testing in various rooms like living rooms, kitchens, bedrooms to ensure the app recognizes markers on different surfaces and under varying lighting conditions.
- Offices and Workplaces: Scenarios involving office equipment, furniture, and variable ambient light.
- Educational Institutions: Classrooms and lecture halls, with a focus on usability for educational purposes.

Outdoor Environments

- Urban Streets: Busy streets with varying lighting and background noise, testing the app's performance in a crowded, dynamic environment.
- Parks and Open Spaces: Natural lighting and different types of natural surfaces, including grass, trees, and water bodies.
- Commercial Areas: Shopping malls, markets, where the app might be used for interactive advertising or navigation.

Lighting Conditions

- Bright Daylight: Ensuring the app works well in direct sunlight.
- Low Light: Testing in evening or dimly lit conditions.
- Artificial Lighting: Various indoor lighting conditions, including fluorescent and incandescent lights.

Physical Surfaces and Markers

- Flat Surfaces: Tables, walls, and floors, testing the app's ability to anchor AR objects.
- Irregular Surfaces: Objects with uneven surfaces, like sculptures or plants.
- Moving Surfaces: Testing with markers on moving objects or people.

User Interactions

- Gestures: Swiping, pinching, and tapping to interact with AR content.
- Movement: Walking around or moving objects to see how the AR adjusts.
- Voice Commands: If supported, testing voice interaction under various ambient noise levels.

Specific Use Cases

- Educational Content: Interacting with educational material in AR, such as historical reconstructions or scientific models.
- Marketing and Advertising: Engaging with AR ads, like interactive posters or product packaging.
- Entertainment: Playing AR games or experiencing AR stories and art.

Accessibility

- For Users with Disabilities: Testing with screen readers, voice navigation, or other accessibility tools.
- Ease of Use for All Ages: Ensuring that the app is user-friendly for both younger and older users.

Network Conditions

1. Wi-Fi Connectivity: Testing app performance on stable, high-speed internet.
2. Mobile Data: Ensuring functionality on various mobile networks.
3. Offline Mode: If applicable, testing how the app performs without internet.

The above scenarios are just a few of the many possible scenarios.

4.2.4. Test Coverage

Considering the technical and time constraints, the most rational approach is to conduct:

Functional Testing, which will include the following checks:

- Marker Detection and Multi-marker Detection: Verify the app's ability to detect and track AR markers in various conditions.
- AR Content Rendering: Test how well the app renders AR content, including 3D models, animations, and interactive elements.
- User Interaction: Assess the app's response to user inputs like touch, gestures, and voice commands.
- Content Management: Test functionalities related to managing AR content, such as downloading, updating, and deleting content.

Also, partially conduct Compatibility Testing - Network Compatibility: Evaluate the app's performance in various network conditions and speeds.

4.2.5. Scope of Testing

a) Functionality Testing

The main objective is to ensure that all features of the Zappar app work as intended and provide a seamless user experience.

Key Areas to Test

Multi-Markers Detection and Tracking: Test the app's ability to quickly and accurately detect and track markers in various environments and lighting conditions.

AR Content Rendering: Evaluate the rendering of AR content, such as animations, 3D models, and interactive elements, ensuring they appear correctly and without delay.

User Interactions: Verify the app's response to user inputs, including touch gestures, swipes, and any other interaction methods supported by the app.

Content Management: Test the functionalities for downloading, updating, and managing AR content within the app.

Test Scenarios:

Scanning markers placed in different positions in a well-lit room and observing the speed and accuracy of the appearing AR content.

Interacting with AR content, like moving or resizing 3D models, and checking for responsiveness and any glitches.

Testing the download and update mechanisms for new AR content and ensuring smooth integration within the app.

b) Compatibility Testing

Objective: To confirm the stability of the application's performance under various network conditions.

Key Area of Testing: Network Compatibility - the app's performance under different network conditions.

Testing Scenario: assessing the app's performance in areas with varying network strength and speed, including testing how well it performs with limited or no internet connectivity.

In both Functionality and Compatibility Testing, the goal is to cover a comprehensive range of scenarios and conditions to ensure that the Zappar app delivers a reliable and high-quality experience to all users, regardless of their device or environment.

4.3. Testing Tools

For conducting the testing, an approach involving the use of a virtual machine and a device with the testing software has been chosen (see fig. 2.4.).

For this, it was chosen:

a) The Android Studio Emulator is a feature within Android Studio, the integrated development environment (IDE) for Android app development. It's a tool that allows developers to simulate different Android devices on their computers. This emulator provides a convenient way to test and debug Android applications in a controlled environment without needing a physical device.

The emulator replicates the functionalities and behavior of various Android devices, including smartphones and tablets. It allows developers to test their applications across different Android versions, screen sizes, hardware specifications, and configurations. This flexibility is crucial for ensuring that apps perform consistently and as expected across the diverse Android ecosystem.

One of the key advantages of the Android Studio Emulator is its deep integration with Android Studio. It provides features like drag-and-drop installation of apps, screen recording, and even simulating different network conditions, GPS locations, and hardware sensors. Developers can also use it to simulate user interactions with the app, including multi-touch gestures, device rotation, and other physical actions.

Additionally, the emulator supports advanced features like OpenGL ES graphics and camera emulation, making it particularly useful for testing more complex applications, such as games or AR apps. Its performance and fidelity in emulating Android devices make it an essential tool for Android developers.

b) Scrcpy is an open-source application that provides a way to display and control Android devices from a desktop computer, whether it be Windows, macOS, or Linux. The name "scrcpy" stands for "screen copy". This tool is highly valued for its performance and low latency, making it a popular choice for a wide range of applications, from app development and testing to gaming and general device management.

One of the key features of scrcpy is that it does not require any root access to the Android device. It works by creating a server on the Android device and then transmitting the screen data to the computer, where it's rendered in a window. The tool also supports sending input from the computer back to the Android device, enabling full control of the device using the computer's keyboard and mouse or touchpad.

Scrcpy is known for its high-resolution and smooth display capabilities, maintaining good performance even at high screen resolutions. It can handle real-time interaction, making it useful for tasks that require rapid response, such as gaming or interactive app testing. Moreover, it's lightweight and doesn't impose significant performance overhead on the device.

Another advantage of scrcpy is its simplicity and ease of use. It doesn't require a complex setup or configuration, and it connects to the Android device via a USB cable or wirelessly. This makes it a convenient tool for developers who need a quick and efficient way to interact with their apps.

c) SplitCam is a software application designed for video streaming and webcam effects. It's primarily used to enhance live video calls, streams, and recordings by adding various effects and features. The software allows users to split their webcam video stream, enabling them to use the same webcam in multiple applications simultaneously, which is a functionality not commonly available in standard webcam software.

The core appeal of SplitCam lies in its ability to add fun and creative elements to video streams. Users can apply different filters, backgrounds, and effects to their video feed, making it popular for personalizing online interactions, whether for casual video chats or professional live streams.

In addition to its webcam splitting and effects capabilities, SplitCam often includes features for screen sharing, recording videos, and streaming to various platforms. This makes it a versatile tool for content creators, gamers, and anyone looking to enhance their live video presence.

SplitCam can be substituted with Webcamoid for higher-quality video transmission, although additional configuration will be necessary.

4.4. Testing tools communication

The interaction can be described as follows: Scrcpy gains control over the device and streams its screen to the computer. SplitCam connects to this display stream and broadcasts the image onto a canvas, which can display images thanks to a virtual camera receiving the display stream.

The Android Emulator is capable of working only with a physical camera, which is usually designated as the default webcam. By disabling the webcam, the virtual camera takes the place of the default camera and can also be used by the Android

Emulator. However, this approach has a significant drawback. When using SplitCam, we cannot control the size or scale of the area transmitted to the emulator, resulting in the image being cropped. This problem can be solved by using Webcamoid because it does not automatically create a virtual camera with fixed characteristics – the camera needs to be manually created using the console by setting the necessary characteristics, which significantly improves data transmission to the emulator but requires additional time for configuration.

It should also be noted that the method of replacing the default camera is usually the most effective way to use a virtual camera. However, it is also possible to try specifying the correct index in the name for the virtual camera within the configuration file. An important aspect is that this method only works with cameras that have a public tag. Additionally, it is worth mentioning that in the MacOS system, there is a possibility to find out the camera's number.

4.5. Preparing for testing tool

Preparation for testing involved setting up the working environment, searching for special images - zapcodes, which are used for AR effects in Zappar, configuring the virtual environment and virtual components, and verifying the functionality of Zappar in the virtual environment.

4.6. Testing the Zappar application

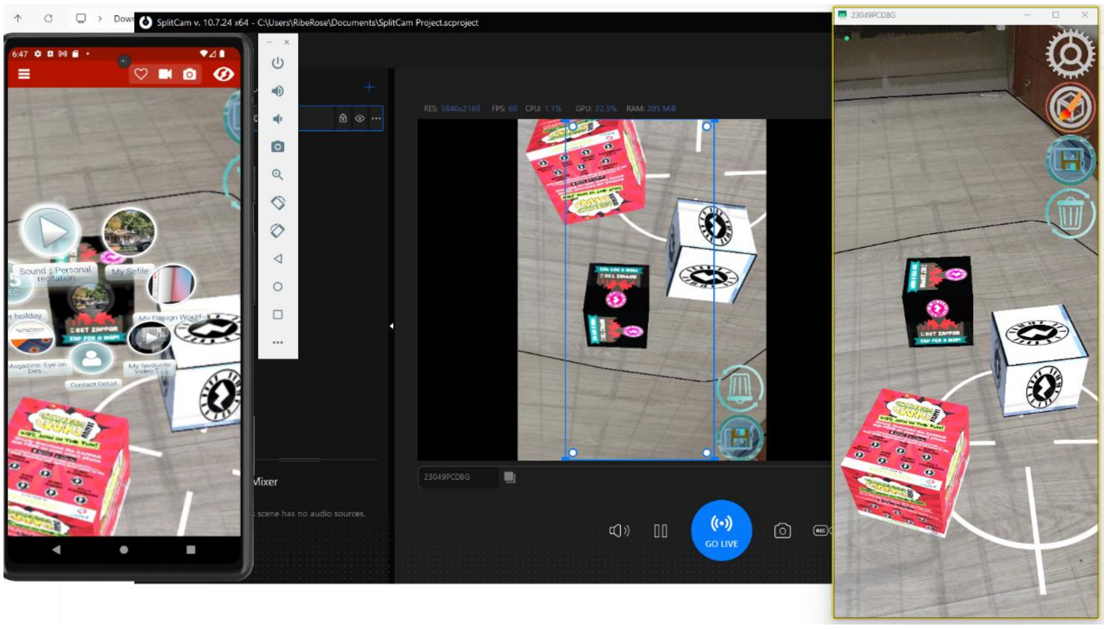
SplitCam v. 10.7.24 x64 - C:\Users\RibeRose\Documents\SplitCam Project.scproject
Mixer
GO LIVE
23049PCD8G

Fig 4.1. Class diagram (part 1)

Testing was fully conducted using the application for testing and the emulator (refer to Fig. 4.1). It was found that only one program can be processed at a time, and objects at a distance were processed first. Additionally, in the creation of 3D objects, they could not interact with the user but were able to interact with objects in the environment. The program was sensitive to lighting and could not function effectively with poor lighting or in its absence.

The results of the test showed:

Lighting Sensitivity: The testing revealed the application's high sensitivity to lighting conditions, indicating a need for optimal lighting for effective AR rendering.

Single Program Processing: The limitation of processing only one program at a time suggests a need for optimization to handle multiple tasks concurrently, enhancing user experience.

Object Processing Order: The preference for distant objects in processing could impact the AR experience, particularly in scenarios where foreground objects are more critical.

3D Object Interactivity: The inability of 3D objects to interact with the user might limit the application's use in interactive AR experiences, though their interaction with environmental objects is a positive aspect.

Recommendations for Improvement: Based on these findings, it's recommended to enhance the application's light processing capabilities, improve multitasking functionalities, refine object processing priorities, and explore ways to enable user interaction with 3D objects.

Further Testing: Additional tests under different lighting conditions and with varied user interaction scenarios could provide more insights into the application's performance and areas for improvement.

User Experience Consideration: Future tests should also consider the overall user experience, especially in scenarios where the user's interaction with the AR environment is crucial.

These observations and recommendations can guide further development and refinement of the application to better meet user needs and improve overall performance.

Conclusion

This section has systematically presented and defined the foundational principles underlying the proposed methodology for testing AR applications. A crucial aspect that emerged is the current lack of standardized AR testing programs. While this absence allows for greater flexibility and freedom in developing testing methodologies, it simultaneously introduces a level of uncertainty. Addressing this uncertainty is pivotal in establishing robust and reliable testing practices for AR applications.

Furthermore, the section elaborated on a specific methodology that utilizes an AR testing tool. This tool plays a crucial role in the interaction of AR programs and facilitates the essential transfer of data between them. The methodology's effectiveness hinges on this interaction and data transfer capabilities, underscoring the need for innovative solutions in AR application testing.

In addition, a brief overview of potential interaction methods was provided, offering insights into various approaches for data transfer to the devices. These methods include cloud-based interactions, the use of virtual machines, and leveraging external computing devices, each with its unique advantages and challenges.

Lastly, a comprehensive description of the methodology and the steps for its implementation was outlined. This detailed account serves as a guide for effectively employing the methodology in practical testing scenarios.

In conclusion, the development and refinement of this methodology represent a significant contribution to the field of AR application testing. As AR technology continues to evolve, the adaptation and enhancement of these testing approaches will be crucial for ensuring the reliability and effectiveness of AR applications in various domains.

Conclusion

In this section, comprehensive and methodical testing of the Zappar software was conducted. The initial phase involved defining test objectives and scenarios, where both functional and non-functional requirements were carefully analyzed. This step was crucial to understanding the specific needs and expectations of the program's core user groups. The scope of testing was carefully outlined, ensuring that all critical aspects of the application, including user interface, functionality, performance and compatibility, were properly covered. By defining the test coverage, the testing process was adapted to comprehensively evaluate the capabilities and performance of the program.

Preparation for testing included setting up the necessary tools and environments for individual interaction. This setup was integral to creating realistic test scenarios that closely mimic real-world conditions and user interactions.

Actual testing of the Zappar application was conducted using an emulator and other relevant tools, which provided valuable information about the application's performance under various conditions. The testing process revealed important findings, such as the app's sensitivity to lighting and its limitations in handling multiple apps at the same time. In addition, it emphasized the dynamics of interaction between 3D objects and the user, as well as with other objects in the environment.

Therefore, the test results provide a solid basis for further optimization and improvement of the application, ensuring that it meets the ever-evolving needs of users and remains competitive in the dynamic field of augmented reality applications.

CONCLUSIONS

During the execution of the diploma project, the fields of augmented reality (AR) application development and testing were explored, and several issues in this area were identified. These issues include the absence of specific tools for conducting testing within the augmented reality sphere and the lack of development and testing standards for software, resulting in an ambiguous situation in this field.

Additionally, a new approach to testing augmented reality applications was proposed within the framework of the diploma project. This approach is based on conducting testing using another AR application, effectively within the realm of mixed reality. Possible implementation options for the interaction of these applications were presented, along with the challenges associated with their implementation. Furthermore, the steps of this methodology were outlined.

To conduct such testing, it was decided to create an augmented reality application that could place objects in the real world, which could then be used for testing purposes. In the second section of this work, the functional capabilities of this application were presented, and its structure was depicted in the form of a class diagram.

The methodology was applied to test the Zappar application. Overall, its functional and non-functional requirements were determined, as well as its target audience and usage scenarios. The scope of testing was defined. During the testing process, certain issues with this program were identified.

Additionally, during testing, it was discovered that many emulators do not support the selection of a camera or a virtual camera, which further complicates the testing process. It was also found that not all virtual machines support augmented reality applications due to the inability to install the required service or due to outdated system configurations or bugs. Furthermore, it was revealed that some applications either do not work at all within an emulator or, although available, do not work properly.

In conclusion, as a result of completing the thesis, a methodology for testing augmented reality software using a mixed environment created by other software has been developed. This approach allows not only to test the program's functionality in a conventional environment but also its operation within a mixed reality context with the emergence of cloud-based augmented reality. This methodology transcends hardware limitations and is becoming increasingly straightforward to use. Additionally, it currently demonstrates the possibility of implementing mixed reality with at least one augmented reality program.

REFERENCES

1. Systematic Systematic Mapping Studies in Software Engineering / P.Kai, F. Robert, M. Shahid, M. Michael // Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering / P.Kai, F. Robert, M. Shahid, M. Michael. – Swindon, United Kingdom, 2008. – (BCS Learning & Development Ltd). – С. 68–77.
2. Style guidelines for naming and labeling ontologies in the multilingual web / [E. Montiel-Ponso, D. Vila-Suero, B. Villazón-Terraz та ін.]. // Dublin Core Metadata Initiative. – 2011. – №11. – С. 105–115.
3. Mobile Indoor Augmented Reality. Exploring applications in hospitality environments. / [B. Barbolla, C. Corredera, J. Ramón та ін.] // 1st International Conference on Pervasive and Embedded Computing and Communication Systems / [B. Barbolla, C. Corredera, J. Ramón та ін.]. – Algarve, Portugal, 2011. – (Science and Technology Publications, Lda). – С. 232–236.
4. Embedded System Architecture for Mobile Augmented Reality. Sailor Assistance Case Study. / [J. Diguet, N. Bergmann, J. Morgère та ін.] // 3rd International Conference on Pervasive and Embedded Computing and Communication Systems / [J. Diguet, N. Bergmann, J. Morgère та ін.]. – Barcelona, Spain, 2018. – (Science and Technology Publications, Lda). – С. 16–25.
5. A Practical Framework for the Development of Augmented Reality Applications by using ArUco Marker / [D. Avola, L. Cinque, G. Foresti та ін.] // Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods / [D. Avola, L. Cinque, G. Foresti та ін.]. – Rome, Italy, 2016. – (Science and Technology Publications, Lda.). – С. 645–654.
6. Liberatore M. Virtual, mixed, and augmented reality: a systematic review for immersive systems research / M. Liberatore, W. Wagner. // Springer Science and Business Media LLC. – 2021. – №3. – С. 773–799.
7. Augmented Reality Learning Experiences: Survey of Prototype Design and Evaluation / [M. Santos, A. Chen, T. Taketomi та ін.]. // IEEE Transactions on Learning Technologies. – 2014. – №1. – С. 38 – 56
8. Rafi T. PredART: Towards Automatic Oracle Prediction of Object Placements in Augmented Reality Testing / T. Rafi, X. Zhang, X. Wang // 37th IEEE/ACM International Conference on Automated Software Engineering / T. Rafi, X. Zhang, X. Wang. – USA, 2011. – (Dublin Core Metadata Initiative). – С. 105–115.
9. Wenkai L. A State-of-the-Art Review of Augmented Reality in Engineering Analysis and Simulation / L. Wenkai, A. Nee. // MDPI AG. – 2017. – №3. – С. 17.
10. Singh S. Analysis of Software Testing Techniques: Theory to Practical Approach / S. Singh, S. Tanwar. // Indian Journal of Science & Technology. – 2016. – №32. – С. 1–6
11. Software Testing: Survey of the Industry Practices / J.Kasurinen, A. Knutas, O. Taipale, T. Hynninen // 41st International Convention on Information and

Communication Technology, Electronics and Microelectronics / J.Kasurinen, A. Knutas, O. Taipale, T. Hynninen. – Opatija, Croatia, 2018. – (IEEE)

12. Guoning Y. Testing of mobile applications. A review of industry practices [Електронний ресурс] / Y. Guoning, Z. Wenkai // Blekinge Tekniska Högskola, Institutionen för programvaruteknik. – 2019. – Режим доступу до ресурсу: http://urn.kb.se/resolve?urn=urn:nbn:se:bth-17880.

13. Khan R. Agile approach for Software Testing process [Електронний ресурс] / R. Khan, A. Srivastava, D. Pandey // 2016 International Conference System Modeling & Advancement in Research Trends (SMART). – 2016. – Режим доступу до ресурсу: https://www.researchgate.net/publication/315914633_Agile_approach_for_Software_Testing_process

14. Vukovic V. A Business Software Testing Process-Based Model Design / V. Vukovic, J. Djurkovic, J. Trninic. // International Journal of Software Engineering & Knowledge Engineering. – 2018. – №5. – С. 701–749.

15. Augmented Reality Issues – What You Need to Know [Електронний ресурс] // The app solutions. – 2023. – Режим доступу до ресурсу: https://theappsolutions.com/blog/development/augmented-reality-challenges/.

16. 3 Challenges Of Augmented Reality Development [Електронний ресурс] // ImagineAR. – 2021. – Режим доступу до ресурсу: https://imaginear.com/blog/ar-development-challenges.

17. 12 Augmented Reality Challenges [Електронний ресурс] // XR. – 2023. – Режим доступу до ресурсу: https://www.xreality1.com/artificial-intelligence/augmented-reality-challenges/.

18. What are the Challenges Faced by AR App Developers? [Електронний ресурс] // ParamInfo. – 2019. – Режим доступу до ресурсу: https://paraminfo.com/what-are-the-challenges-faced-by-ar-app-developers/.

19. Explore The Challenges and Opportunities of Developing AR/VR Solutions [Електронний ресурс] // A3logics logo. – 2023. – Режим доступу до ресурсу: https://www.a3logics.com/blog/explore-the-challenges-and-opportunities-of-developing-ar-vr-solutions.

20. Dafnis C. Applications Analyses, Challenges and Development of Augmented Reality in Education, Industry, Marketing, Medicine, and Entertainment [Електронний ресурс] / C. Dafnis, L. David, P. Luis // mdpi. – 2023. – Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/13/5/2766.

21. Watson T. SPECIFICS AND CHALLENGES OF AUGMENTED REALITY TESTING [Електронний ресурс] / Tracy Watson // Skywell Software. – 2019. – Режим доступу до ресурсу: https://skywell.software/blog/specifics-and-challenges-of-augmented-reality-testing/.

22. Martis B. The 7 QA Software Testing Principles [Електронний ресурс] / Beniamin Martis // Linkedin. – 2022. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/7-qa-software-testing-principles-beniamin-martis.

23. Quality Assurance, Quality Control and Testing — the Basics of Software Quality Management [Електронний ресурс] // Altexsoft. – 2018. – Режим доступу до

ресурсу: https://www.altexsoft.com/whitepapers/quality-assurance-quality-control-and-testing-the-basics-of-software-quality-management/.

24. Venkatesh V. 7 Principles of Software Testing [Електронний ресурс] / Vasu Venkatesh // Linkedin. – 2023. – Режим доступу до ресурсу: https://www.linkedin.com/pulse/7-principles-software-testing-vasu-venkatesh.

25. Shah H. The Key Principles of Software Testing Every QA Must Consider [Електронний ресурс] / Hardik Shah // Able.bio. – 2021. – Режим доступу до ресурсу: https://able.bio/hardikshah/the-key-principles-of-software-testing-every-qa-must-consider--02b0618d.

26. How do you measure and improve the performance and reliability of AR and VR applications? [Електронний ресурс] // Linkedin – Режим доступу до ресурсу: https://www.linkedin.com/advice/0/how-do-you-measure-improve-performance-reliability#testing-metrics.

27. Minor S. Test automation for augmented reality applications: a development process model and case study [Електронний ресурс] / Sascha Minor // Degruyter. – 2023. – Режим доступу до ресурсу: https://www.degruyter.com/document/doi/10.1515/icom-2023-0029/html.

28. How do you create effective AR test cases? [Електронний ресурс] // Linkedin. – 2023. – Режим доступу до ресурсу: https://www.linkedin.com/advice/3/how-do-you-create-effective-ar-test-cases-skills-augmented-reality.

29. Uddin A. Importance of Software Testing in the Process of Software Development [Електронний ресурс] / A. Uddin, A. Anand // International Journal for Scientific Research & Development. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/331223692_Importance_of_Software_Testing_in_the_Process_of_Software_Development.

30. Dynamic Testing Techniques of Non-functional Requirements in Mobile Apps: A Systematic Mapping Study / [M. Júnior, D. Amalfitano, L. Garcés та ін.] // ACM Computing Surveys / [M. Júnior, D. Amalfitano, L. Garcés та ін.]. – New York, United States, 2022. – (Association for Computing Machinery New York, NY, United States). – С. 1–38.

31. Okezie A. A Critical Analysis of Software Testing Tools / A. Okezie, I. Odun-Ayo, B. Sherrene. // Journal of Physics Conference Series. – 2019. – №4. – С. 1–11.

32. Augmented Reality: Survey [Електронний ресурс] / [E. Carlos, J. Carlos, L. Santos та ін.] // Mdpi. – 2023. – Режим доступу до ресурсу: https://www.mdpi.com/2076-3417/13/18/10491.

33. What is Quality Assurance (QA) in Software Testing? [Електронний ресурс] // Testsigma – Режим доступу до ресурсу: https://testsigma.com/guides/quality-assurance/.