Journal of Artificial Intelligence General Science (JAIGS)

ISSN: 3006-4023 (Online), Volume 6, Issue 1, 2024 DOI: 10.60087

Home page https://ojs.boulibrary.com/index.php/JAIGS

Transforming Software Testing in the US: Generative AI Models for Realistic User Simulation

S A Mohaiminul Islam[1], MD Shadikul Bari[2], Ankur Sarkar[3]

[1]Master of Science in Information Technology, Washington University Of Science & Technology (WUST), Alexandria, Virginia, USA mohaiminulbd271@gmail.com
[2]Master of Science in Information Technology, Washington University Of Science & Technology (WUST), Alexandria, Virginia, USA shadikulbari@gmail.com
[3]Master of Science in Information Technology, Washington University Of Science & Technology (WUST), Alexandria, Virginia, USA ankursylhet@gmail.com

Abstract

Testing software has a higher level of difficulty because of the variations in users' behaviors, decreasing time of software development, and the demand for prototypical testing. It becomes almost impossible to apply traditional approaches in determining the software's dynamic environment or its ability to capture different users' interactions. Introducing to this paper is the hybrid model of Generative AI and RL to model realistic user behaviors whilst modulating to software responses as well. Specifically, in this paper, we discuss the US context by tackling several challenges specific to the regional context of the demographic diversity, the widespread use of Agile/DevOps methodologies and frameworks, and the demand for the highest levels of quality in software testing. The combination of Generative AI for behavior variety with RL for learning makes the given methodology a continuous feedback process for the sake of thorough and realistic behavioral testing. This is well illustrated by real-life applications in areas like e-commerce, healthcare and banking to mention but a few where the model provides robust results terms of identifying difficult to detect faults, test effectiveness and cost benefit analysis. They plan to co-designing federated learning for privacy-preserving testing in the future, as well as leveraging more cross-cultural user simulations that have global application.

Keywords: Generative AI, Reinforcement Learning (RL), User Simulation, Software Testing, US QA Landscape

ARTICLE INFO: *Received: 19.10.2024 Accepted: 10.11.2024 Published: 15.12.2024*

Introduction

Software testing QA process is one which focuses on checking the functionality, performance, and usability of an application before its release. However, growing customer engagement interfaces, partly due to demographic, gadget, and geographic variation, present immense issues to conventional testing approaches. Especially in the US, where the variability of user behavior is relatively high, there is a strong need for highly accurately reproducing usage scenarios at the simulation level.

1. Problem Statement and Unique Angle

1.1. Challenge in User Simulation

The first and major limitation of the user simulation is that user models are static, while users and the software changes over time. Almost all current simulation approaches work by assuming average users or even scripting typical user interactions when, in fact, users behave more randomly in practice. This limitation results in a substantial disconnect between the simulated scenarios and real-world usage, ultimately undermining the effectiveness of software testing such as:

a) **Inability to Adapt to Software Changes:** Current user behavior simulation techniques lack flexibility, which means that there is no ability to change quickly according to changes in the software functions or upgrades. When there are modifications in the software applications such as new features, user interfaces or systems, static user simulation models fail to communicate the effect of change on the users. Therefore, software testers work with only simulations that may not be proper to current world uses or may not in any way mirror a given user's experience. This lack of flexibility can cause large risk in quality assurance because testers cannot understand the behavior of real users on changes in the modified software.

b) **Difficulty in Capturing Complex User Interactions:** Secondly, the conventional techniques often prove ineffective in capturing instances that may occur during the use of the software, including complex user engagement patterns. It is well documented that there are many determinants to user behaviour such as attitudes, goals, context and mood. Traditional approaches tend to model users in a rather restricted way thus providing only rather limited simulation that does not take into account user irregularity. Thus, during testing phases, software seems to work just fine, while this is not quite true. When the product is in the real environment, there are always scenarios the designers never considered during the testing process, thus, an undesirable user experience and possible product dissatisfaction.

1.2. US-Specific Context

Current situation of software testing in USA is cited as being highly diverse and this makes it challenging for QA squads. In the case of their American counterparts, which this writer has interacted with, the behavior of users is inversely proportionate to numerous determinants of social factors mainly age, gender, socio-economic class, and culture. This kind of user engagement generates multiple vectors that define the challenges for software developers and testers when reacting to an ever-more diverse audience's expectations. Some of the conventional testing approaches that can involve a substantial use of test users with predefined patterns of activity cannot reveal this diversity. As such, it becomes very hard to avoid software failure when using it in live conditions; this greatly decreases the satisfaction of the end user and may lead to a company suffering major reputation losses.

Growing focus on the testing of realistic user behavior can be viewed as essential in this regard. A product must also not be overachieving only in controlled environments, but also in actual interaction scenarios with other methods or potentially unpredictable opponents. In light of the demographic differences across

the United States testing has to target the different device, operating system and network environments from city to rural areas. The problem is that the traditional approaches to testing cannot handle this variation well enough, which reduces the usefulness of the testing results for assessing the performance of the system.

The climate in software development has radically changed due to the implementation of Agile and DevOps approaches. These paradigms focus on velocity, integration on a frequent basis, and rapid cycle times, and therefore revolutionize the conventional approaches toward testing the software. There is today a growing demand for organisations to provide products that not only work properly but also perform predictably and dependably in light of variability from users. Therefore, there is a need to develop testing techniques that can mimic real users' activities in an intelligent and efficient manners and that adapt to changes in users' behavioral patterns.

1.3. Proposed Unique Approach

The foundation of this study is built upon the application of Generative AI models, including but not limited to GANs or diffusion models, which are used to effectively model numerous types of user engagement. The scripts used in traditional forms of user simulation are prescriptive, which reduces the possible contexts that can be assessed. On the other hand, Generative AI models can generate multiple behaviours of users, which means that the Mimic can capture how actual users can interact with software applications in a richer way.

The strengths of the Generative AI model in their work, therefore, are its capability to be trained using datasets and to generate new data that is different but similar in characteristics. This capability is used to obtain atypical user interactions which might have been omitted to be observed, thus providing a diverse dataset of unpredictible human interaction. Thus, mimicking such various behaviors, software testing can concern the situations and peculiarities that are crucial for the high reliability and efficiency of software in practical usage. It will be worth to note that both GANs and diffusion models are capable of being updated perpetually, which makes them fit for this function. Their generative processes are capable of emulating the interactions as seen from differ ent users possessing differing backgrounds, skills and usage patterns.

The second platform of this combined approach is the Incorporation of Reinforcement Learning to improve the manners in which Generative AI models simulate user behaviors. RL adapts systems by allowing them to learn what actions reward them, relative to punishments based upon information obtained from an environment. When integrating of RL into the user simulation framework, the technology allows the modification of the simulated user behaviours depending on the responses offered by the software.

This learning mechanism emerging in the process is essential for extending a mining search path to resemble that of a human user, as a human subject does not follow linear model, but rather, learns from his experience and tends to change strategies consequently. For example, if a simulated user perceives an error or an undesirable outcome when testing a software application, then RL can instruct the simulated user to change their bahavor in a successive test. This completes the feedback loop so that both the Generative AI Model and the software that is being tested will improve periodically. It enables the interactive and dynamic coordination of the model through which it is possible to emphasize some pathways, test other behaviors and, overall, lead to better testing results.

The integration of Generative AI and RL creates a context where the simulation is not only occurring with users, but also is reflecting the learning processes that occur with users. Consequently, testing extends beyond merely being a validation activity; it becomes a sensible evaluation of software behavior and provides insights and chance for improvement.

2. **Data Collection:** Performance of AI based any model very much depends upon the quality and the data collected from the relevant source. In software testing, it is necessary to mimic the real user environment, meaning that recorded interactions have to reflect as much as possible the actual usage of the application.

For our research, we will focus on two primary sources of data: submissions of real application usage logs from users in the United States, and user heat maps and actual session usage reports.

2.1. **Behavioral Data Sources:** Application logs shall be the primary data since they contain a record of behaviors as customers interact with e-commerce apps, social media, and banking. Collecting logs from these applications will provide an abundance of information, including but not limited to:

a) **User Actions:** Each time the user interacts on the page, every click, every scroll or every input a timeline is generated. This data is vital for training the Generative AI in a way it recreates genuine user action data since it emulates the users' behavior observed while using this app.

b) **Session Duration and Frequency:** Knowing how often users use an application over some time helps in determining the amount of user loyalty and level of interest. Such information assists in constructing user profiles that might be helpful for the RL part of the envisaged model.

c) **Error Logs:** Not only do we want to note good interactions, but it is also very important to know where the users fail. Reviewing the error logs will let us model different situations where a user might get upset or quit a task, and this, in turn, the AI model will consider.

d) **Demographic Data:** While we will exclude personal identifiers to address concerns of legal compliance, others like age, geographical location, and devices where the content will be consumed enhance the richness of our information. It enables segmentation during model training and an analysis of the diverse behavior of users across the various strata.

Together with the application logs, heatmap offers an effective visualization of the interaction of users with the interface. Maps highlight what is clicked, moused over, and scrolled on a Web page or application screen, displaying the parts most engaged. The utilization of heatmaps serves several critical functions in our methodology:

a. **Identifying User Interests:** Heatmaps shed light on the areas of an interface that generate the most enthusiasm, in which case, where clients find most interest or engagement within interface elements or content. Such knowledge can help the Generative AI model generate incorporation of realistic use preferences and favorable navigation patterns.

b. **Optimizing User Interface (UI) Design:** Heatmap insights are useful for redesigning a site's UI as businesses pay attention to areas which the consumers seem to be having confusion or low interaction with. This data will be used in our research to model potential user annoyance and learn potential behavioral adjustments in the RL model that will create a feedback loop of simulated user annoyance as well as of enhanced actual software experience.

c. **Session Recording Data:** In contrast with simple heat maps, session recordings depict interactions as a sequence of events during the user's session with the application. This includesities, pauses, and impro- visations in the sequence of event from the standard protocol. Such density of information is often essential for fine-tuning simulations as well asboarding the context of a user.

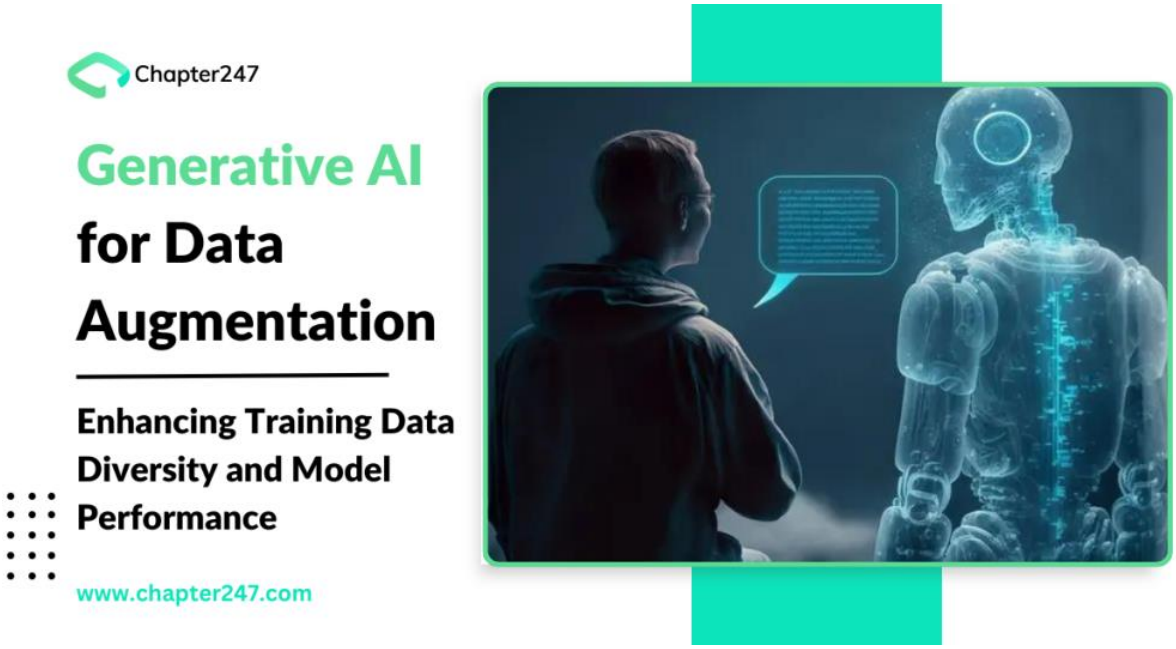
2.2. Synthetic Data:

Synthetic data therefore can be described as artificially created data that is modeled to represent real life data in order to allow the testing of various aspects without having to use real user data or large amounts of it to achieve this goal. The incorporation of synthetic data into software testing is no more a luxury but a strategic imperative owing to the shortcoming of conventional means of data gathering that lacks ability in modeling complex real-world behavioural scenarios which are less frequent but critical alignments or from some minority streams of users.

a) Generative AI for Data Augmentation

In our first approach, we use generative AI methods, such as GANs and VAEs, to create realistic synthetic data. Specifically, in the context of applying software testing, synthetic data appears as a supplement to the datasets normally used in testing. The generative models use existing user behavior datasets as the base to train from as the synthetic behaviors produced should mirror realistic usage patterns in addition to capturing relatively rare usage patterns.

Figure 1



Source: Chapter247 Infotech

- **Training the Generative Model:** The first stage is to build a strong Generative model with the help of the user interaction data collected in the past. This training utilizes numerous data sources that can reflect a broad range of activities of users, usage logs, feedback and interaction rates. The above/types of training data can assist the generative model to learn from typical as well as non-typical users' behavior.

- **Synthetic Data Generation:** After training, the generative model allows for new synthetic user interactions. This includes using the trained generative model to generate new sequences of the target user behaviour which mimic the exceptional and extreme cases. For example, it can generate sequences depicting disabled users in interaction with the interface, abnormally patterned usage, or when the interface interacts with atypical hardware. The generated scenarios will comprise user intent, environmental variables as well as context of use as well as unusual behavior cases.

- **Augmentation of Testing Data:** The final objective is to extend the current pool of data by the data that is produced by the system to expand the spectrum of testing possibilities. Through synthetic data,

software testers are able to determine the ability of their applications to perform under such conditions which are not particularly common in standard user interactions. This augmentation though is necessary when it comes to testing frameworks for application, to test to what extent an application is insensitive to varying user inputs.

b) Addressing Underrepresented Behaviors

With respect to this research, first, underrepresented behaviors are of most importance, second, the behaviors that seldom happen, the so-called rare edge-cases, and, third, accessibility. Most conventional testing datasets were designed with formal user interactions in mind, leaving out the additional challenges of software usage by a variety of users disabilities, or other conditions.

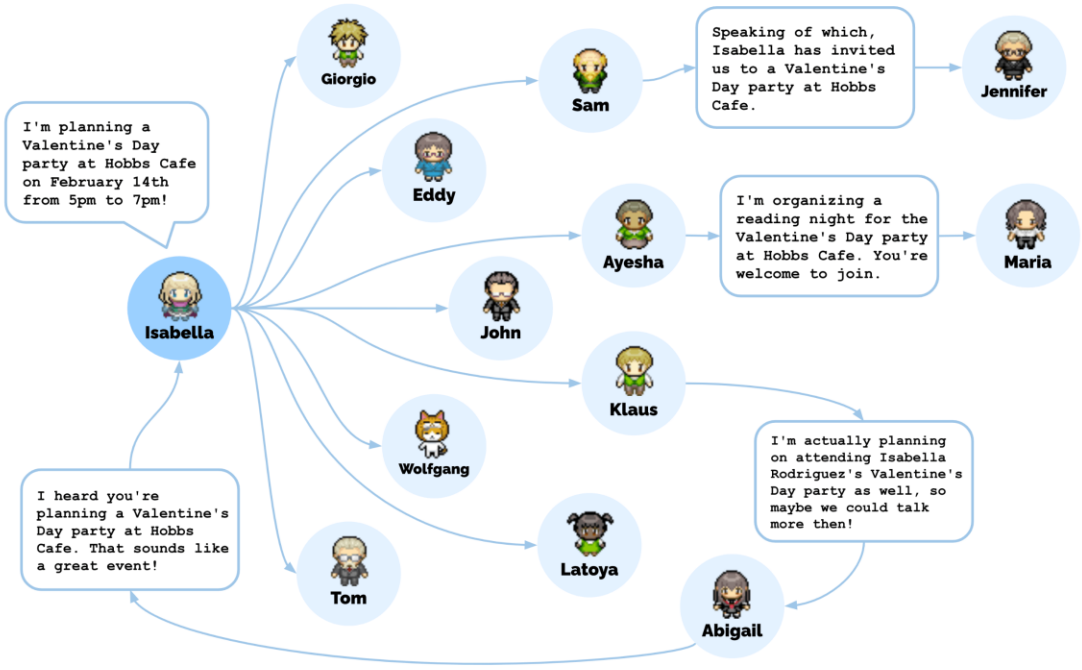
- **Rare Edge-Case Scenarios:** These scenarios generally are not included or poorly represented in testing data, for the simple reason that the conditions where they take place rarely are encountered in practical settings. However, they can cause significant changes in software performance. For example, problems in which user launches a function in rather specific settings or through certain interfaces can be rather revealing. In this context, to create a realistic environment within which such events can be tested, artificial data are used that reproduces such exotic cases and therefore improves the evaluation of the software in terms of its preparedness to address any unexpected input.

- **Accessibility Testing:** These scenarios generally are not included or poorly represented in testing data, for the simple reason that the conditions where they take place rarely are encountered in practical settings. However, they can cause significant changes in software performance. For example, problems in which user launches a function in rather specific settings or through certain interfaces can be rather revealing. In this context, to create a realistic environment within which such events can be tested, artificial data are used that reproduces such exotic cases and therefore improves the evaluation of the software in terms of its preparedness to address any unexpected input.

- **Feedback Mechanisms:** The combination of reinforcement learning with…generative AI enables one to have a more or less dynamic approach towards synthesized behaviors. Reflecting the findings of real-world testing related to user interaction, RL algorithms of the system constantly evaluate the overall performance; as a result, the generative model is redirected for further targeted synthetic data generations. This repetitive process not only introduces new variants into the dataset but also provides relevance and accuracy of information synthesized from the generated user interactions.

2.2 Generative AI for User Behavior Simulation

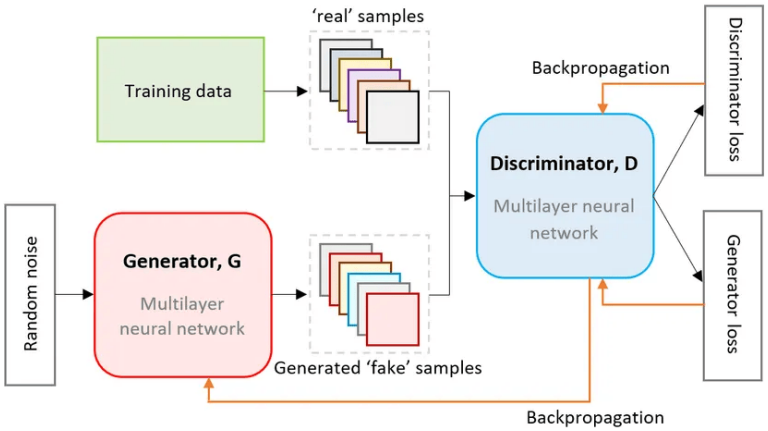
Generative AI in User behaviour simulation as an Application for software testing is a progressive and radical notion. Moreover, it makes use of other kinds of machine learning like GANs and VAEs as a way to model realistic, diverse and contextually relevant user behaviors. These simulations are not limited to text-based, passive, or pre-stored scenarios but effective dynamic prototypes to check whether these software systems are optimized to the best possible usability for people with disabilities.
Figure 2.



Source: Generative AI Simulate Human Behaviour

2.2.1 Generative Adversarial Networks (GANs): Based on an unsupervised learning paradigm, GANs come as two neural networks: The generator and the discriminator which work in somewhat antagonist fashion regarding the objective. The generator synthesizes data points like user interactions while the discriminator judges how close these point are to actual users' behavior. These adversarial interactions help to hone the generator's capability for the construction of progressively more authentic behavior patterns in due course.

Figure 3.



Source: Exploring the Power of Generative Adversarial Networks (GANs)

2.2.2 Variational Autoencoders (VAEs): VAEs map high-dimensional user data into a lower-dimensional space, importantly preserving key aspects, while keeping freedom for synthesizing various behaviors. While GANs are used for generating specific lookalikes of completed transactions, VAEs aim at approximating the required probabilistic distribution of user behavior for differentiating between demographic-specific or other unexpected activities.

2.3 Simulating Diverse User Profiles

Category	Sub-category	Description	Examples

Behavioral Patterns	Clicks and Swipes	Simulating sequences of clicks, gestures, and swipes based on realistic user interactions.	Context-based clicks, gesture-based swipes
	Navigation Paths	Developing realistic user navigation through software, including typical workflows and random patterns.	Standard user processes, random path exploration
	Interaction Timing	Modeling the timing between actions to reflect true human behavior in software.	Simulating realistic delays between user actions
Demographic-Specific Behaviors	Age-Based Variations	Variations in behavior based on age, affecting interaction speed and error rates.	Younger users navigate faster; older users may pause or retry more often
	Geographic Preferences	Regional differences in user preferences, such as language, interface layout, and menu options.	Location-based language selection, menu placement preferences
	Device Usage Patterns	Simulating interactions for different devices (screens, touch interfaces, keyboards) and connection speeds.	Mobile vs. desktop interaction differences, adapting to screen size
Unexpected User Actions	Input Errors	Testing software's response to incorrect or non-meaningful user inputs.	Wrong email format, unsupported characters
	Edge Case Interactions	Deliberate attempts to trigger insecure or unintended behaviors through unusual interactions.	Selecting out-of-bounds options, clicking disabled buttons
	Non-Standard Navigation Paths	Using unconventional navigation sequences, revisiting completed stages, and exploring hidden paths.	Reaching target page via atypical routes, revisiting previously completed steps

2.4. Integration of Reinforcement Learning

The concept of utilizing RL in conjunction with a Generative AI and RL system changes software testing by providing intelligent user emulation performed by the RL algorithm. Controlling and modulating user actions during the testing process is one of the critical aspects of RL agents designed to respond to the SUT by tuning/altering users' behavior in subsequent iterations. This section discusses how RL agents help in improving the quality, range, and density of users' behavior modeling in the testing scenario. Using RL Agents to refine and adapt user behavior, the hybrid architecture, Reinforcement Learning agents are central. These agents model themselves as testers by engaging themselves of the target SUT and evaluate their behavior via feedback loop in real-time facilitated by Machine Learning (ML). Their functionality revolves around two critical areas: using the examples of software reactions and choosing less explored scenarios.

2.4.1. Learning from Software Responses: It should be noted that RL agents are especially effective in analyzing the system's reactions and replying in turn. Key feedback sources include:

a) **Loading Delays:** Whenever it is detected that the SUT is performing poorly, RL agents follow up by performing stress tests to determine capacity limits or choke points, areas where SUT perform worst.

b) Error Messages: Administrators for example look at error codes or exceptions that come with them, agents reproduce the contexts in which these error messages came up.

c) Unexpected System Behavior: If the SUT does not behave as expected (e.g., unresponsive UI elements, or abnormal outputs), RL agents purposely induce these conditions repeatedly and uniformly to fully test and verify the defect.

It does so in order to mimic real-world usage patterns emphasizing the ability to adapt to varieties within a given type.

2.4.2. Exploring Uncommon Paths

Generative AI often models ordinary and expected user interactions with a given system in consideration to past occurrence or likely future cases. Nevertheless, situations that occur usually, specific procedures that are utilized sparingly, or incidental usage of multiple aspects by different user types are typically overlooked. RL agents address this limitation by:

a) **Identifying Gaps:** They also study some parts of the application where communications are rare or possibly have not been tested at all, and proceed to guide the simulations into those regions.

b) **Intelligent Randomization:** For Generative AI models, RL agents bring in new input patterns, or navigation paths, which are quite outside the training data given to Generative AI models.

c) **Complex Scenarios:** They use a sequence of two or more workflows or actions to confirm the program's versatility when facing different scenarios.

This capacity for 'new frontiers' guarantees a higher and wider testing range, thus a minored risk of bugs going through to live.

2.5. Reward System:

Like most of the RL agents depend heavily on the design of the reward scheme. This system drives the agents to align their actions with the overarching goals of software testing: identifying of faults, achieving thorough test, and mimicking of real-life behavior of the application. Below are the reward agents for:

2.5.1 Discovering previously undetected bugs:

a) Those that were not previously discovered in prior test cycles are especially useful. For such defects, RL agents receive very high rewards, which promotes new and creative approaches to testing.

b) Incentives depend on the type of defect that has been found with critical bugs being those that cause crashes or corrupt data being given the highest rewards

2.5.2 Exploring untested path:

a) Paths through the software not scripted for testing are a big plus.

b) Incentives are attached as au higher remunerations for response to trends of work flows or conditions that were hitherto ignored.

2.5.3 Simulating realistic behaviors leading to high-impact defects: Some simulations that produce serious consequences such as software crashes when emulating logical user activity receive enormous bonuses. It also maintains the extant that the generated behaviors are both realistic and effective.

2.6 Implementation Strategies

a) **Multi-Agent Systems:** The testing process benefits from using a set of RL RL agents with different goals. For instance, one agent may deal with a performance testing while another agent deals in security testing vulnerability.

b) **Collaborative Learning:** Multi-agent RL progress is achieved by agents exchanging experiences on their test sessions, and the results narrow the gap to more effective tests.

c) Interface with Generative AI

- ✓ **Initial Inputs:** The generative AI gives reference directions of how the user will engage with the system and the real-life situations learned from past or from the given specialism.
- ✓ **Dynamic Refinement:** Chou and colleagues argue that RL agents modify and extend these scenarios depending on updates, forming a loop of constant improvement.

d) **Reward Shaping for Further Development:** Variable rewards help agents in gradually improving the testing approaches over time and a step by step improvement of testing efficiency.

2.7 Feedback Loop: Generative AI + RL

Also, the integration of Generative AI and Reinforcement Learning (RL) in software testing is progressed by incorporating a reliable feedback loop on board. This loop enables an ever improving influence of the simulated behaviours, whereby the system will keep on improving its state as the tests go on. The loop promotes a two-way relationship between Generative AI and RL given that Generative AI will generate realistic and diverse user simulation while RL will make intelligent adjustments to the generated behaviors considering the way Real Software interacts with it.

Components of the Feedback Loop

This is an information feedback process involves Generative AI and Reinforcement Learning in an interchanging manner in a cyclically manner. This process is outlined in three core stages:

2.7.1 Behavior Simulation with Generative AI

In the context of this work, generative AI is used as the basis for designing various scenarios of users' interaction. By leveraging historical user data, domain knowledge, and probabilistic modeling, it generates:

A) **Common User Patterns:** Developed using particular usage patterns, which can be related to the way a user clicks through a menu, completes a form or interacts with specific application segments.

B) **Rare or Edge-Case Behaviors:** Operations resembling sporadic use, for example, rapid clicks that do not correspond to any objective interaction or the usage of multiple input keys at the same time or after a long period of no activity.

C) **Scenario Variations:** Variations in input timing, sequences, and environmental conditions, multiple potential for different combinations of the successive workflows.

Descriptive AI offers equal generic sets of synthetic user actions as a basis for testing, since generative AI delivers the roster of potential interactions.

2.7.2 Adaptive Refinement with RL

After the behaviors are generated, RL agents then intervene to further optimize these simulated actions according to the feedback from software interfaces and testing results. Key processes in this stage include:

a) **Dynamic Adaptation:** Developed RL strategies check the experienced behaviours on the SUT and adapt their actions proceeding from the reaction. For example:

- ✓ If the loading of agents experiences delays, more stress tests can be carried out in similar types of workflows.
- ✓ Whenever there is an error message, agents look for the neighboring functionalities or input variants to get into the problem and examine the defect.

b) Exploration of Untested Areas: RL agents engage certain facets of application that Generative AI simulations may not have explored thoroughly, let alone certain regions that are now considered its edges.
c) Feedback-Driven Learning: RL agents monitor the extent to which specific behaviors lead to results, such as finding bugs, evaluating performance characteristics or limitations, or revealing vulnerabilities.
This ensures that the behaviour simulated at this stage change gradually in line with experiences from real life environments, with a view of making them much more relevant and effective.

2.7.3 Retraining Generative AI with RL Insights

The last step is based on the RL agents and the process of feeding back, the information acquired by these agents back into the Generative AI models. This re training assures that the Generative AI is updated in accordance to the needs of testing environment. Key aspects include:

a) **Incorporating New Defect Patterns:** Information gathered by RL agents is used to create promising instances that caused defects to occur in more subsequent testing.

b) **Enhancing Realism:** Timing perturbations or navigation choices that have been learned by RL agents are incorporated into Generative AI as a way of enhancing the realism of future runs.

c) **Expanding Coverage:** The generative AI models are trained additionally to accommodate the RL agents with new-use cases and enhanced the data coverage in terms of a range of user behaviors.

By updating the Generative AI, the feedback loop enables an iterative learning model in which each cycle enhances the quality and potency of the modeled users.

2.7.4 Key Benefits of the Feedback Loop

The last step is based on the RL agents and the process of feeding back, the information acquired by these agents back into the Generative AI models. This re training assures that the Generative AI is updated in accordance to the needs of testing environment. Key aspects include:

A. **Incorporating New Defect Patterns:** Information gathered by RL agents is used to create promising instances that caused defects to occur in more subsequent testing.

B. **Enhancing Realism:** Timing perturbations or navigation choices that have been learned by RL agents are incorporated into Generative AI as a way of enhancing the realism of future runs.

C. **Expanding Coverage:** The generative AI models are trained additionally to accommodate the RL agents with new-use cases and enhanced the data coverage in terms of a range of user behaviors.

By updating the Generative AI, the feedback loop enables an iterative learning model in which each cycle enhances the quality and potency of the modeled users.

2.7.5 Implementation Strategies for the Feedback Loop

The last step is based on the RL agents and the process of feeding back, the information acquired by these agents back into the Generative AI models. This re training assures that the Generative AI is updated in accordance to the needs of testing environment. Key aspects include:

a) **Incorporating New Defect Patterns:** Information gathered by RL agents is used to create promising instances that caused defects to occur in more subsequent testing.

b) **Enhancing Realism:** Timing perturbations or navigation choices that have been learned by RL agents are incorporated into Generative AI as a way of enhancing the realism of future runs.

c) **Expanding Coverage:** The generative AI models are trained additionally to accommodate the RL agents with new-use cases and enhanced the data coverage in terms of a range of user behaviors.

By updating the Generative AI, the feedback loop enables an iterative learning model in which each cycle enhances the quality and potency of the modeled users.

3. Experimentation

The blended Generative AI and RL framework is an acceptable solution setting when dealing with the complexities in the software testing process. As applied to actual and real users who behave in certain ways the usage of this method can greatly enhance the speeds and efficiency of SQA as it includes the aspect of real life feedback under testing simulation. This section presents a discussion of the strategies of applying this approach, and the result of a pilot experiment conducted in the context of the US software testing environment.

3.1. Implementation in US Software Testing Context

To demonstrate the effectiveness of this hybrid approach, we focus on three representative domains: These are e-commerce, healthcare, and banking. The focus with these examples remains on the wideness of applicability, though special attention is made to the domain-specified opportunities that stem from using the hybrid form of the model.

3.1.1. Case Studies:

Application	Objective	Testing Scenarios	
E-commerce Platform	Test for usability and transactional reliability under various user conditions.	- **Cart Abandonment**: Users navigate through products, add to cart, but do not complete checkout. - **Frequent Page Switching**: Rapid switching between categories to test session management. - **Invalid Coupon Entries**: Enter expired or incorrect coupon codes.	Detects issues affecting user retention, conversion rates, and financial trans
Healthcare App	Ensure robust handling of sensitive operations and user accessibility.	- **Invalid Data Inputs**: Test boundary values for health-related data. - **Slow Internet Speeds**: Test app performance under varying network conditions. - **Accessibility Testing**: Use screen readers, voice	Identifies weaknesses in accessibility, input validation, and performance under load.

		commands, and keyboard emulation for compliance checks.	
Banking App	Ensure security and reliability during sensitive financial operations.	- **Frequent Session Timeouts**: Test session management by simulating frequent inactivity. - **Incorrect Authentication Attempts**: Model repeated failed login attempts to assess fraud detection and account security.	Enhances the security, session management, and error handling mechanisms critical to financial operation

- **E-commerce Platform**

a. **Objective:** Log in with various and plausible users to look for issues involving usability and such financial transactions.

b. Scenarios:

1. **Cart abandonment:** Multiply users who just navigate through products, add something to the cart but do not complete the check out before exiting the session.
2. **Frequent page switching:** Users which can also switch quickly between the categories, thus unmasking the problems with session management or caching.
3. **Invalid coupon entries:** Case scenarios at which customer seek to use the coupon code after its expiry or using the wrong coupon code.

c. **Impact:** It also recognizes bugs that regulate the user retention, number of conversions in the sales funnel, and the number of correct transactions.

- **Healthcare App**

a) **Objective:** Check that applications allowing key sensitive operations are well buffered and tested with respect to user accessibility and errors encountered.

b) Scenarios:

1. **Invalid data inputs:** Use negative and bound cases like submitting values involving health that may be at his/her extreme low or high.
2. **Slow internet speeds:** It is important to test the application under different network environment in order to ensure that it will perform well in terms of satisfying the need of the underserved areas.
3. **Accessibility testing:** Use keyboard emulation and screen reader and voice command to check for accessibility to standards such as ADA or WCAG.

c) **Impact:** Identify weaknesses in features of accessibility, validation of the input, and performance during conditions that may include load.

- **Banking App**

a) **Objective:** Laboriously guarantee that security concerns remain intact during operations such as authentication as well as transactions processing.

b) Scenarios:

1. Frequent session timeouts: Imitate users who, after a number of inactivity, have their session terminated time and time again.

2. Incorrect authentication attempts: Model users typing incorrect credentials several times to check the effectiveness of the frauds detection and account locking.

Impact: Enhances reliability of the security, session management and error handling mechanisms that are so vital for financials.

3.1.2. Tools & Frameworks: To implement the hybrid Generative AI and RL approach effectively, the following tools and frameworks are utilized:

A. Generative AI Training:

- ✓ **TensorFlow or PyTorch:** For training models to mimic real users with realistic behaviors in a reasonable amount of cycles. These frameworks enable defining the interaction dynamics of a user, using pre-trained models, such as GPT or diffusion models.

B. Integration with Testing Pipelines:

- ✓ **Selenium or Appium:** Use it to perform real user scenarios on web or mobile apps allowing for automated testing. These tools allow integration with other platforms and bring out the specific UI issues.

C. Reinforcement Learning for Adaptation:

- ✓ **OpenAI Gym:** Enables a dynamic behavior based on RL in the environment. RL agents can recognize from the software responses the result they get during testing and adjust the relevant simulated behaviors to be closer to the emerging real-life patterns.

D. **Supporting Infrastructure:**

- Web based environments such as AWS or Google Cloud to extend training and testing environment.

3.2. Metrics for Evaluation

In order to confirm the hybrid approach, we use KPI based on the aspects of realism, efficiency, coverage, and appropriateness.

a. Realism

- ✓ **Approach:** As a next step, the simulated behaviors have to be compared with the actual user audit information (such as the clickstream data).

- ✓ **Metric:** The average percentage of simulated interaction to corresponding user behavior patterns.

b. Bug Detection Efficiency

- ✓ **Approach:** Quantitative comparison of the bugs detected by the hybrid model and the bugs discovered by other techniques.
- ✓ **Metric:** Number of high priority defects found in relation to a number of test cycles.

c. Coverage

- ✓ **Approach:** Evaluate the richness of the testing matrix with regard to the coverage of the specific areas and the performances of the system under various conditions, primarily focusing on the marginal and composite states and interactions.
- ✓ **Metric:** Proportion of the novel scenarios of the system use, which are often neglected in the manual or automated tests, identified and incorporated into the framework.

d. Adaptability

- ✓ **Approach:** Determine if the RL component increases the performance of simulated behaviours in response to software changes, work flows or any other circumstances.
- ✓ **Metric:** Delay in the time for which RL is initiated to continue the performance levels in new conditions where it is implemented.

4) Addressing US QA Industry Needs

QA plays an important role of delivering high quality applications which meet the needs of users in various industries. The concerned challenges that the US QA industry has experienced are: Testing for cultural and dynamically diverse users, establishment of the integration of the QA processes with the Agile/DevOps methodology, and testing large-scale, complex systems. This paper discusses how the use of Generative AI as a user modeling technique for emulating users' behaviors and RL as a technique for modifying the system's actions in response to the users' behaviors can overcome these issues.

This method simulates the interactions from typical users with Generative AI according to the demography and the device type. In the same process, RL further fine-tunes these simulations based on responses made by the software used in that endeavor and in doing so stays current and more appropriate. Below, I provide further an elaboration on how this form of hybrid staffing meets the necessary needs of the US QA industry.

4.1 Diversity of User Base

The population density in the US and the cultural, technological and demographic diversification of the region insist on a broad spectrum in the user behaviors that has to be tested by the software testing. Thus, the hybrid model demonstrates proficiency in these variations and guarantees absolute test coverage.

4.1.1Demographic Differences

- ✓ **Urban Users:** Such users utilize well-endowed applications with better internet connection and tend to do multiple activities at the same time they interact. Some of the emulated behaviors are often switching between contexts, using multiple applications at once, and usage of quick navigation patterns.

- ✓ **Rural Users:** Customers at the rural level may encounter slow connectivity, operate with fewer steps in work processes, and may engage with applications for more time. The includes on and off connectivity, slower input rate and very limited multitasking.
- ✓ **Benefits:** This demographic-aware simulation helps review areas with performance problems, UI/UX incompatibilities, and resource consumption precluding high satisfaction in various locations.

➢ **Age Groups**

- ✓ **Younger Users:** As one subculture familiar with technology they are inclined to gamified interfaces, frequent and intensive interactions and most specifically such innovative enhancements as augmented interfaces. The sighted behaviors are things like fast feature navigation, purchasing for products inside an application, and personalization.
- ✓ **Older Users:** These users easily concern themselves with simplicity, easy access and clear forms of designs. The specific aspects of simulations are slow movement, accessibility features such as a bigger font or a speaker that reads the text aloud, and erroneous input.
- ✓ **Benefits:** Testing helps to support the cause of inclusiveness, as it provides verification of the likelihood of application accessibility within all age brackets, especially as regards compliance with the necessities of the elderly.

4.1.2. Device Preferences

➢ **Mobile Devices**

- ✓ Engage in mimicry of touch activities, back, front, / top/bottom screen position swapping and battery demanding actions.
- ✓ State problems related to layout, recognition of gestures, and measures for increasing work efficiency, respectively.

➢ **Desktop Devices**

- ✓ Specific events include, but not limited to, multitasking with multiple tabs, keyboard operations, or screen resolutions.
- ✓ Correct the incompatibility of browsers and proper division of resources during intensive operations.

➢ **Tablet Devices**

- ✓ Integrative Mobile and Desktop – unite stylus inputs, split-operation, and hydraulic navigation.
- ✓ Make it work smoothly on different shape and sizes.

Impact: Continual testing not only across the different platforms but also across the different types of devices guarantees the satisfaction for every user preferring a given device.

4.2. Agile/DevOps Integration

The hybrid model incorporates effectively with Agile and the DevOps as these practices focus on constant deployment practices and evolvement related with software development life cycle. It also improves its capability to integrate with CI/CD pipelines; improving the real-time testing and flexibility.

4.2.1 Approval for addition to Standard CI/CD pipelines

- **Continuous Testing**

During each build or release different simulated user behaviors are performed together with automated test suites. This leads to the point that any new code is compared with its performance in real-life usage scenarios as soon as possible.

RL techniques maintain the optimum level of simulation learning dependent on previous-testing-a/B-testing insight or analytical data.

Outcome: The ability of moving along the progressive flow more quickly and the possibility of detecting defects before regression in rapidly growing applications.

- **Real-Time Adaptation**

RL agents keep track of the software reaction and the reported errors and delays and modify the emulated actions to hit the vulnerable areas.

Example: If a feature detecting high error rates exists, the RL model exposes intensive simulation on related workflows to identify problems.

Outcome: Adaptive testing helps to stay relevant especially when a development environment is constantly fluctuating with new features being added.

4.2.2. Tools for Integration

- **Generative AI:** TensorFlow and PyTorch to design complex users' behavior patterns.

  - ✓ **Testing Frameworks:** Selenium, Appium, and JUnit for getting the results of simulations in different platforms.

  - ✓ **RL Frameworks:** AI implementation in the form of OpenAI Gym to change test case based on the real-time response received.

4.3. Scalability for US Market

System level applications in the present day U.S. applicative terrain including e-business trawlers, health care structures, and finance services call for volumetric testing to guarantee dependable and satisfactory performance. The architecture of the hybrid model can be seen to correspond to these demands.

4.3.1. Handling Complexity

- **E-commerce**

  a) Mimic actions like cart abandonment, high-traffic product sale events, and many forms of the payment system.

  b) Recognize problems related to checkouts, validation of promotional codes, and the manner, in which systems can be scaled, when there is incredible traffic.

4.2.2. Healthcare Systems

a) Perform different testing scenarios following the usage of HIPAA compliant data, emergency response and multi-device compatibility.

b) Guarantee high dependability rates in conditions like high influx of ER data and multiple telemedicine sessions.

- **Financial Applications**

a) Model safe, frequently processed commercial transactions, including users' personal information.

b) Check the effectiveness of fraud fighting tools and the reliability of the system when the market is volatile

4.3. Creating Large Scale Simulation Capability

- **Cloud Deployment**

a) Use of AWS, Azure or Google Cloud to deal with huge calculations processes.

b) Emulate numerous user models for practical, training, and execution purposes on different distributed systems environments for millions of users.

- **Parallel Execution**

Leverage the distributed computing concepts to run a number of what-if models more than the one environment.

Impact: Mass simulation makes certain that applications can run on high volumes of users, complex activities and numerous situations without slowing down.

4.4. Met out the Needs of the US Market

- **Efficiency:** These realistic dynamism mimicry rapidly and simultaneously increases the eficiency of the QA phase while not sacrificing its comprehensiveness.
- **Cost-Effectiveness:** Lower hardware dependency and cost of operations are possible when the cloud approach is used.
- **Market Relevance:** Specific models for US users allow every software to provide product services for heterogenic population in United States.

Help to address the issue of high complexity and volumes of applications that are in the large-scale software testing environment in the USA.

5) Benefits of the Unique Approach

Introducing Generative AI and Reinforcement Learning (RL) into software testing brings a fresh and revolutionary perspective that proposes to solve problems that QA faced for a long time. This integrated model offers the necessary dynamic flexibility, better detection of defects, and a closer relation to the users' actual patterns all with regard to costs. In the table below, it is broken down where each of these benefits represent a singular, critical element of this fresh approach to technology.

5.1. Dynamic Testing

Another speceficity of the Generative AI and RL combines is dynamic testing, which remains fundamentally important since software constantly improves. Most testing methodologies can fall short in a fast-paced development process, in which features and graphical interfaces are likely to evolve.

5.1.1. Key Dynamic Testing Affairs

- **Behavioral Evolution:** Generative AI develops different and realistic approaches of users. RL modifies such models in response to the software reactions or find the ways for the simulated behaviors' evolution connected with the application.

Example: The hybrid model of e-commerce platform testing adjusts its simulation as new features such as one-click purchasing or live chat support are added.

- **Continuous Relevance:** Since the static test case is a manual technique, one has to update frequently; however, in the hybrid model, the testing scenario is comprised of automated adaptation.

Example: This allows dynamic test to be integrated on CI/CD on environments that operate with Agile/DevOps architectures.

5.1.2.Impact

- Improvement of coverage on new and modified features with little intervention from the manpower.

- Minimization of possibilities of having concealed flaws due to the usage of outmoded test models.

5.2. Uncover Rare Defects

Probably the biggest difficulty in software testing is the detection of the hard to find, or marginal errors which can cause huge problems in critical domains such as financial and medical. The hybrid model is thus well suited to solve this challenge.

5.2.1.How the Hybrid Model Recognises Unique Defects

- **Crucial Generative AI's Broad Scenarios:** There is the ability of generative AI in mimicriing almost every real and imaginary user interaction with technological products.

Example: It verifies issues like, what happens if someone inputs multiple invalid medical codes at once? Or extreme app slowdown when doing demanding calculations?

- **RL's Targeted Exploration:** RL algorithms centers on stress points and boundary conditions in response to software. These are concentrated where possible lacks are anticipated, increasing exposure of defects.

Example: For banking application, RL might perform a series of failed authentications then a correct one in order to analyze how it behaves under stress.

5.2.2.Impact:

- Better definition of high-risk deviations that might cause concern such as system breakdowns, security compromises or infringement of some regulatory requirements.

- Improvement of levels of dependability of applications in the sorts of businesses as finances, medical care, and online marts, and that is due to the fact that edges always prove to be fatal.

5.3. Real-World Alignment

Exercise regarding user behavior should be conducted to ensure that the final developed software corresponds to the user's expectations. The approach guarantees that actual interactions by the users are

closely simulated especially by conducting the simulation in the diverse context of the United States market.

5.3.1. Key Features of Real-World Alignment

- **Demographic-based Simulations:** Generative AI considers input data from users of different background; the younger and elder generations, those residing in urban and rural areas and those with different levels of computing literacy.

Example: They can emulate an urban millennial user often with fast internet connection on a smartphone or a senior rural user often with a slow desktop connection.

- **Behavioral Accuracy:** Since training the model involves the processing of analytics and feedback data, the model mimics realistic operations and workers' habits- multitasking, making mistakes or using assistive technologies.

Example: In the case of a retail app, the model may mimic leaving a cart, scrolling through the list of products reviews, or applying irrelevant coupons.

- **Continuous Feedback Loop:** Actual user data improves the simulation and the process makes the model to become more real as the progress continues.

5.3.2 Impact

- Applications are pre- scrutinized for conditions that users are expected to find in real life hence minimizing the disparity between the testing and actual production environment.

- Increased user satisfaction as software is best placed to deal with the realities of the world.

5.4. Cost-Effective Testing

Another strength of the hybrid model is the ability of the model to avoid unnecessary excessive expenses. Typical testing methodologies involve large amounts of time devoted to manual testing and engaged resources that puts a lot of pressure on the QA budget. The hybrid model eliminates or minimizes these aspects as it automates and optimises the testing process.

5.4.1. How the Model Reduces Costs

- **Automating the process of Behavior Adaptation:** RL reduces the time required to replicate the test scenarios against a new release of the software to a greater extent than when primarily using test cases.

Example: Whenever a financial app is embarking on adding a new feature, such as investment tracking, the model will include related test cases in its design making it less work for the developers.

- **Decrease in actual human involvement:** Testers employing generative simulations and automated learning apply their efforts to high value-added activities such as defect reviews and not generative and learning scenarios.

Example: A healthcare appliances application QA team can focus its effort on verifying compliance requirements while the model is testing for boundary conditions and cross-platform compatibility.

- **Efficient Resource Management:** Cloud-based execution of simulations is modeled in a manner which enables its on-demand scalability: no expensive infrastructure to support.

5.4.2. Impact

- Substantial reduction of the QA labor costs and expenses required for infrastructure.

- Reduced time to market: Testing cycles that are accelerated but do not neglect product quality.

6. Conclusion

Combined Enabling technologies of Generative AI and Reinforcement Learning (RL) appear as the solution to the testing dynamics of software in the US market focusing on issues of user variability, dynamic software specifications, and the optimization of the testing strategies.
The hybrid system helps with user modeling by creating realistic model simulations with Generative AI that stand for the pluralism of the US market. This ensures the application has its SOS tested while in a state that is as close as possible to how the application will be used. Moreover, the model's adaptability by RL enables regular iterations, whereby the simulated actions are modified based on outcome experiences and software modification.

6.1. Key Transformations:

6.1.1 Addressing User Diversity

- Since different users are simulated within the hybrid model, including rural users, urban users or different age groups, everyone is accommodated.
- Simulations at device level are optimal for rigorous testing on mobiles, desktops, and tablets and others.

6.1.2. Increasing Testing Efficiency

- Constant run adaption automatically reduces the extent to which engagement is needed which in turn fosters time and effort conservation while achieving maximum test coverage.
- It is possible to customize dynamic testing capabilities to adjust for evolving software because Agile/DevOps predominates the development process.

6.1.3. Enhancing Defect Discovery

- Due to the identification of such exceptional situations and glitches, the hybrid model can be considered as effectively suitable for industries such as finance and healthcare.

This novel approach changes the way how basic QA techniques are performed and how they can be adjusted to better respond to practical needs. As a result, addressing the specific needs of the US market in this case provides a new reference for testing software.

6.2. Future Work

Although much progress is made by adopting the current hybrid model, the future studies can go even further with extending the next innovative dimensions of the business.

6.2.1. Technique for Secure Data Analysis at the Edge: Federated Learning

Given increasing data privacy related issues, and particularly in the US, federated learning offers a compelling solution such as under GDPR and CCPA. This approach also allows training of models across various distributed data without the need of moving user data to a centralized location.

- **Benefits for QA:**
  - ✓ Users' data can be multi-real in simulation systems and their privacy remains guaranteed.
  - ✓ Decentralized training improves the reliability and accuracy of the simulations that are derived from a range of behavioral patterns.
- **Applications**

- ✓ **Healthcare:** Protect patients' information when using apps to model such use cases as HIPAA and secure telemedicine.
- ✓ **Finance:** Preserve privacy in imitating a user's behaviors to detect fraud and provide secure transactions.

Therefore, by incorporating resilient learning into the hybrid model, the simulations of software testing can meet privacy standards optimally while continuing to be highly realistic.

6.2.2. Real-Life Example of Cross Cultural User Simulation for International Usage

- ➢ **Cross Cultural User Simulation for Global Applications:** In many software companies based in the USA, testing methodologies need to incorporate culturally unique behavior and preferences while catering to the technology needs across the world.
  - ✓ **Expanding Simulated Behaviors:** Here features of cultural differences, for example, in navigation, language choice, or preferred type of payments.

Create device usage behavior tailored to specific areas familiar with the first and the second degrees, such as increased mobile use or work-based desktop use in emerging nations.

- ➢ **Challenges and Solutions:**
  - ✓ **Challenge:** Collection of data that will be used for training models within culturally different areas.
  - ✓ **Solution:** Get in touch with the regional data suppliers while respecting privacy through federated learning.
- ➢ **Global Testing Platforms:** Build efficient paradigms for testing applications within different geographical locations to effectively respond to cultural differences existing in other parts of the world apart from America.

Impact: After simulation can cross cultural, US-based companies will be in a position to develop applications that are optimized for the globe hence improving the usability of the applications and increasing market share.

6.2.3. (Developing Architectures of Hybrid Models)

- ➢ **Incorporating Multimodal Learning:** Further develop simulations to take in multimodal inputs meaning the ability to use voice, text, as well as images to test entire sequences and tasks in smart ways within AI rapid applications.

Example: A smartphone based smart home assistant app might be tested for simultaneous voice, text and mobile app instructions.

- ➢ **Scalability with Quantum Computing:** Seamlessly utilize quantum technology to train and implement Generative AI and RL in the shortest time possible.
- ➢ **Applications:** Operations involving key systems such as the health systems or even the overall national economical systems.

Final Thoughts

The integration of Generative AI and RL is recognized as the new generation of software testing, making certain that Apps are resilient, smart and customer-oriented. As it is applied to the US market today, further modifications in federated learning, cross-cultural simulation, and integration with other technologies can be extended to any other country.

Thus, assimilating these novelties can both improve the grocery quality and accessibility for the clients as well as open the gates to the future intelligent, privacy-oriented, and easily scalable testing approaches for the entire QA industry. This vision makes sure that software is constantly being developed to the modern requirements of user needs in this ever-shifting technological environment.

References

1. Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024). Generative ai. *Business & Information Systems Engineering*, *66*(1), 111-126. https://doi.org/10.1007/s12599-023-00834-7

2. Baidoo-Anu, D., & Ansah, L. O. (2023). Education in the era of generative artificial intelligence (AI): Understanding the potential benefits of ChatGPT in promoting teaching and learning. *Journal of AI*, *7*(1), 52-62. https://doi.org/10.61969/jai.1337500

3. Gozalo-Brizuela, R., & Garrido-Merchan, E. C. (2023). ChatGPT is not all you need. A State of the Art Review of large Generative AI models. *arXiv preprint arXiv:2301.04655*. https://doi.org/10.48550/arXiv.2301.04655

4. Lv, Z. (2023). Generative artificial intelligence in the metaverse era. *Cognitive Robotics*, *3*, 208-217. https://doi.org/10.1016/j.cogr.2023.06.001

5. Euchner, J. (2023). Generative ai. *Research-Technology Management*, *66*(3), 71-74. https://doi.org/10.1080/08956308.2023.2188861

6. Li, Y. (2017). Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274*. https://doi.org/10.48550/arXiv.1701.07274

7. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, *11*(3-4), 219-354. http://dx.doi.org/10.1561/2200000071

8. Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26-38. DOI: 10.1109/MSP.2017.2743240

9. Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M. (2016). Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*. https://doi.org/10.48550/arXiv.1611.05763

10. Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2023). Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1-118. http://dx.doi.org/10.1561/2200000086

11. Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, *21*(2), 178-192. https://doi.org/10.1287/ijoc.1080.0305

12. Orso, A., & Rothermel, G. (2014). Software testing: a research travelogue (2000–2014). In *Future of Software Engineering Proceedings* (pp. 117-132). https://doi.org/10.1145/2593882.2593885

13. Bochmann, G. V., & Petrenko, A. (1994, August). Protocol testing: review of methods and relevance for software testing. In *Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis* (pp. 109-124). https://doi.org/10.1145/186258.187153

14. Baresi, L., & Pezze, M. (2006). An introduction to software testing. *Electronic Notes in Theoretical Computer Science*, *148*(1), 89-111. https://doi.org/10.1016/j.entcs.2005.12.014

15. Ciortea, L., Zamfir, C., Bucur, S., Chipounov, V., & Candea, G. (2010). Cloud9: A software testing service. *ACM SIGOPS Operating Systems Review*, *43*(4), 5-10. https://doi.org/10.1145/1713254.1713257

16. Garousi, V., & Mäntylä, M. V. (2016). A systematic literature review of literature reviews in software testing. *Information and Software Technology*, *80*, 195-216. https://doi.org/10.1016/j.infsof.2016.09.002

17. Sterman, J. D. (1987). Testing behavioral simulation models by direct experiment. *Management Science*, *33*(12), 1572-1592. https://doi.org/10.1287/mnsc.33.12.1572

18. Sterman, J. D. (1987). Systems simulation. Expectation formation in behavioral simulation models. *Behavioral science*, *32*(3), 190-211. https://doi.org/10.1002/bs.3830320304

19. Dutton, J. E., & Stumpf, S. A. (1991). Using behavioral simulations to study strategic processes. *Simulation & Gaming*, *22*(2), 149-173. https://doi.org/10.1287/mnsc.31.7.900

20. Dutton, J. E., & Stumpf, S. A. (1991). Using behavioral simulations to study strategic processes. *Simulation & Gaming*, *22*(2), 149-173. https://doi.org/10.1177/1046878191222001

21. Wang, G., Salles, M. V., Sowell, B., Wang, X., Cao, T., Demers, A., ... & White, W. (2010). Behavioral simulations in mapreduce. *arXiv preprint arXiv:1005.3773*. https://doi.org/10.48550/arXiv.1005.3773

22. Lundström, C., & Lindvall, M. (2023). Mapping the landscape of care providers' quality assurance approaches for AI in diagnostic imaging. *Journal of digital imaging*, *36*(2), 379-387. https://doi.org/10.1007/s10278-022-00731-7

23. Karakhanyan, S., & Stensaker, B. (2020). External quality assurance: The landscape, the players and developmental trends. In *Global trends in higher education quality assurance* (pp. 11-36). Brill. https://doi.org/10.1163/9789004440326_002

24. Kalman, L. V., Lubin, I. M., Barker, S., Du Sart, D., Elles, R., Grody, W. W., ... & Zehnbauer, B. (2013). Current landscape and new paradigms of proficiency testing and external quality assessment for molecular genetics. *Archives of Pathology and Laboratory Medicine*, *137*(7), 983-988. https://doi.org/10.5858/arpa.2012-0311-RA