Automated Testing of Tezos Blockchain-Oriented Software

Afef Jmal Maâlej and Achraf Weli
ReDCAD Laboratory, National School of Engineers of Sfax, University of Sfax, BP 1173, 3038 Sfax, Tunisia

Keywords: Automated Software Testing, Test Results Analysis, Secure Smart Contracts, Tezos Blockchain.

Abstract: This research centers on the testing of smart contracts within the Tezos blockchain as a security verification and validation activity, supporting thus its development life cycle. This specific blockchain type is recognized for its self-modifying feature, which facilitates protocol upgrades without network splits. Despite Tezos advanced technology, smart contracts may still harbor bugs and vulnerabilities, necessitating rigorous software testing for quality and security assurance. The primary aim of this work is to develop a solution addressing the divide between technical blockchain development and non-technical participants in the smart contract ecosystem. Our ST2A testing tool, realized for SmartPy-developed smart contracts, offers a user-friendly platform catering to individuals with limited blockchain or programming knowledge. Its overarching objective is to illustrate the Tezos smart contract testing process, ensuring accessibility, comprehension, and actionable insights for non-developers such as project managers, security auditors, and business stakeholders.

1 INTRODUCTION

Software testing of blockchain-oriented applications plays a crucial role in ensuring the integrity, security, and reliability of these novel software systems. Given the unique characteristics and complexities of blockchain technology, evaluation and testing methodologies must be developed to address the specific challenges and requirements inherent in blockchain applications.

One of the primary focuses of software testing in the context of blockchain is the validation of smart contracts. Smart contracts are self-executing agreements embedded within the blockchain, and thorough testing is essential to identify vulnerabilities, logic flaws, and potential security risks. Test scenarios must be designed to cover a wide range of contract conditions, including edge cases, to ensure their accurate and secure execution.

Additionally, testing the consensus mechanisms of a blockchain is critical. Consensus algorithms, such as Proof of Work (PoW) or Proof of Stake (PoS), ensure the agreement and validation of transactions across the network. Testing these mechanisms involves simulating various network conditions, evaluating their performance, and assessing their resilience against potential attacks or failures.

In summary, software testing of blockchain-oriented applications involves specialized methodologies and approaches specifically tailored to the unique characteristics of blockchain technology. The focus includes validating smart contracts, testing consensus mechanisms, performing integration and security tests, and evaluating performance aspects. Organizations can increase confidence and trust among users and stakeholders by establishing blockchain systems through extensive tests to ensure their robustness and dependability.

In this paper, we consider Tezos as a blockchain platform that offers a unique set of characteristics and advantages such as self-amendment, formal verification, on-chain governance, scalability, performance, and interoperability. In this context, we propose ST2A, a tool for automated testing of smart contracts on the Tezos blockchain. The fundamental aim of our solution is to overcome the difficulties of the testing procedures associated with Tezos smart contracts, rendering them accessible, comprehensible, secure, and actionable for non-developers, including project managers, security auditors, and business stakeholders. In particular, the application streamlines the testing process through automation, enhancing efficiency and user experience.

The remainder of this paper is organized as follows. Section 2 provides basic concepts about the Tezos blockchain and software testing. Section 3 draws a literature review on software testing of Tezos blockchain-oriented applications. Afterwards, we de-

Maâlej, A. J. and Weli, A.
Automated Testing of Tezos Blockchain-Oriented Software.
DOI: 10.5220/0013291000003928
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 541-548
ISBN: 978-989-758-742-9; ISSN: 2184-4895
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

scribe our proposed solution, the SmartPy Test Analysis Application (ST2A), in Section 4. Then, a case study of "TicTacToe" Tezos smart contract testing is presented in Section 5. Finally, in Section 6, we conclude with a summary of our contributions and identify prospective study areas for the future.

2 BASIC CONCEPTS

In this section, we provide an introductory overview of both Tezos blockchain technology and software testing, contextualizing their significance within the broader field.

2.1 Tezos Blockchain Technology

Blockchain is a distributed ledger technology that enables transparent, secure, and decentralized storage and transmission of information. It operates on a network of participants, known as nodes, who validate and record transactions in a distributed ledger, commonly referred to as a public ledger (Feng, 2023).

Fundamentally, the blockchain functions by grouping transactions into blocks, which are subsequently added to a chain of blocks in chronological order. Each block contains a set of transactions along with a cryptographic hash link to the previous block, creating an immutable and tamper-resistant structure (Gupta et al., 2023).

In this work, we consider Tezos blockchain technology (tez, 2023) as a novel approach to software engineering, it presents a unique set of characteristics and advantages. Here are some of the key characteristics and advantages of the Tezos blockchain (Olivieri et al., 2023):

- **Self-Amendment:** One of the distinctive features of Tezos is its ability to self-amend through on-chain governance. This means that the protocol can evolve and upgrade itself without requiring hard forks or contentious community debates. Tezos stakeholders can propose and vote on changes, ensuring a more decentralized and adaptable blockchain ecosystem.
- **Proof-of-Stake (PoS) Consensus:** Tezos utilizes a proof-of-stake consensus mechanism, where validators, known as "bakers", are chosen to create new blocks and secure the network based on their stake in the system. PoS consensus ensures energy efficiency, as it does not require intensive computational power like Proof-of-Work (PoW) systems, and encourages stakeholder participation and security.

  - Delegated Proof of Stake (DPoS): Within the Tezos ecosystem, stakeholders possess the capability to delegate their validation rights to other participants without relinquishing ownership. This sophisticated system fosters a more democratic and decentralized approach to the mining process.

  - Liquid Proof-of-Stake (LPoS): Representing an advancement beyond DPoS, the LPoS mechanism in Tezos empowers token holders to engage in the consensus process as either delegators or validators, thereby contributing to the heightened security of the network.
- **Formal Verification:** Tezos incorporates formal verification, a technique used to mathematically verify the correctness of smart contracts and protocol specifications. By employing formal methods, Tezos aims to enhance security, prevent bugs, and ensure the integrity of the platform, making it more robust and trustworthy.
- **On-Chain Governance:** Tezos features on-chain governance, enabling stakeholders to actively participate in the decision-making process regarding protocol upgrades, parameter adjustments, and the allocation of resources. This inclusive governance model promotes decentralization, fosters community engagement, and avoids the need for contentious hard forks.
- **Smart Contract Platform:** Tezos serves as a platform for developing and executing smart contracts. It supports multiple programming languages, including Michelson, a Domain-Specific Language (DSL) specifically designed for formal verification of smart contracts. This flexibility and security-oriented approach make Tezos an attractive choice for building Decentralized Applications (DApps) and deploying complex smart contracts.
- **Scalability and Performance:** Tezos aims to address scalability challenges through its design and governance model. By allowing stakeholders to propose and vote on protocol upgrades, Tezos can potentially implement scaling solutions to improve network performance and throughput, ensuring the blockchain can handle a growing number of transactions.
- **Interoperability:** Tezos is designed to be interoperable, allowing it to interact and exchange information with other blockchain networks and decentralized systems. This interoperability enables seamless integration with external protocols, facilitating cross-chain interactions and fostering a broader blockchain ecosystem.

These characteristics and advantages of the Tezos blockchain make it an innovative and promising platform for decentralized applications, governance, and secure smart contract execution. Its self-amendment capability, proof-of-stake consensus, formal verification, on-chain governance, and focus on scalability contribute to its potential as a robust and adaptable blockchain infrastructure (Olivieri et al., 2023).

2.2 Software Testing

Software testing forms a vital aspect of the software development lifecycle, guaranteeing the quality, reliability, and functionality of software systems. It entails a methodical examination, evaluation, and validation of diverse components and functionalities to detect defects, errors, and vulnerabilities. The fundamental objective of software testing is to identify potential issues and discrepancies at an early stage of development, enabling developers to rectify them before releasing the software to end-users (Koskinen, 2023).

The necessity for software testing arises due to the inherent complexity of software systems, which encompass interdependent modules, sophisticated algorithms, and intricate user interactions. As software applications gain increasing significance in critical domains like finance, healthcare, and transportation, the importance of rigorous testing practices becomes ever more apparent. A solitary software failure or defect can result in severe consequences, including financial losses, compromised data security, and even endangering human lives (Sugianti et al., 2023).

Software testing includes a broad spectrum of activities, including test planning, test design, test execution, defect tracking, and test result analysis. It employs diverse techniques such as black-box testing, white-box testing, and gray-box testing to evaluate the software's functionality, performance, usability, and security. Additionally, various types of testing, such as unit testing, integration testing, system testing, and acceptance testing, are performed at different stages of the software development process (Zheng, 2023).

The evolution of software testing has been propelled by advancements in testing methodologies, tools, and automation frameworks. Testers employ both manual and automated testing approaches to ensure comprehensive coverage and maximize efficiency. Furthermore, agile development methodologies and DevOps practices have revolutionized the testing landscape, emphasizing continuous integration and continuous testing to support frequent software releases and updates (Sugianti et al., 2023).

2.3 Software Testing of Tezos Blockchain-Oriented Applications

Software testing of Tezos blockchain-oriented applications in particular ensures the integrity, security, and reliability of these specialized software systems. Considering the characteristics of Tezos blockchain technology, testing methodologies must be proposed to address the specific challenges and requirements existing in these applications.

One of the primary objectives of software testing in the context of Tezos blockchain is the validation of smart contracts. Smart contracts are self-executing agreements embedded within the blockchain, and thorough testing is essential to identify vulnerabilities, logic flaws, and potential security risks. Test scenarios must be designed to cover a wide range of contract conditions, including edge cases, to ensure their accurate and secure execution.

Additionally, testing the consensus mechanisms of a blockchain is critical. Consensus algorithms, such as Proof of Work (PoW) or Proof of Stake (PoS), ensure the agreement and validation of transactions across the network. Testing these mechanisms involves simulating various network conditions, evaluating their performance, and assessing their resilience against potential attacks or failures.

Besides, integration testing is another crucial aspect of Tezos blockchain software testing. As blockchain systems typically interact with external systems, such as wallets, exchanges, or other blockchain networks, it is essential to verify the seamless integration and interoperability of these components. Integration tests validate the proper flow of data, the accurate processing of transactions, and the adherence to established standards and protocols.

In addition, security testing is of utmost importance in Tezos blockchain-oriented applications. This involves conducting vulnerability assessments, penetration testing, and code reviews to identify potential security loopholes or weaknesses in the blockchain network and associated applications. Thorough security testing helps mitigate the risk of malicious attacks, unauthorized access and data breaches.

Furthermore, performance testing is also significant in the context of Tezos blockchain applications. The ability of the blockchain network to handle a high volume of transactions, scalability, and responsiveness is essential to ensure its practical usability. Performance tests simulate real-world conditions and stress the system to evaluate its response time, throughput, and overall performance under varying loads.

To sum up, software testing of Tezos blockchain-oriented applications involves specialized approaches made to the unique characteristics of blockchain technology. The focus includes validating smart contracts, testing consensus mechanisms, conducting integration and security testing, besides evaluating performance aspects. By ensuring the reliability and robustness of Tezos blockchain systems through comprehensive testing, organizations can build trust and confidence among users and stakeholders.

3 RELATED WORKS

In this section, an exhaustive analysis delves into various methodologies applied to test blockchain-based applications, with a specific emphasis on those rooted in Tezos technology.

In their research, (Rahouti et al., 2023) present a blockchain-centric strategy for visual navigation, meticulously crafted for a heterogeneous team of robots navigating an extensive visual domain. Noteworthy is its elimination of the dependence on map data structures, catering to robotic platforms with limited computing resources. The approach leverages real-time visual information, fostering resilient path selection and adaptive navigation. Additionally, a streamlined Proof-of-Work (PoW) mechanism is introduced to enhance consensus within the untrusted visual network.

Expanding on this, (Arts et al., 2023) articulate the use of Property Based Testing (PBT) techniques to automate testing of fundamental components of the Aeternity blockchain, ensuring its robustness and high-quality performance. PBT, though potent, poses challenges in the intricate task of testing blockchains. The Aeternity property-based testing model aligns with the blockchain structure, systematically separating diverse functionalities into distinct model sections, enhancing clarity, and reducing boilerplate test code. The focus revolves around identifying valid blockchain transactions, with meticulous instrumentation enabling a comprehensive observation of feature combinations and their frequencies during tests. This documentation offers insights applicable not only to Aeternity but also to the testing of other blockchains and complex feature-based systems.

Moving forward, (Nishida et al., 2022) delineate the Helmholtz type-based static verification tool for Michelson, a statically typed stack-based language for Tezos smart contracts. Built on an extended Michelson type system enriched with refinement types, Helmholtz stands as a robust tool for verification. It scrutinizes a Michelson program's type adherence to a user-defined specification, offloading verification conditions to the SMT Z3 solver. Helmholtz's refinement types cover the foundational Mini-Michelson calculus, incorporating complex features like compound data types and higher-order functions. Its efficacy is demonstrated through successful verification of diverse Michelson programs.

Transitioning to broader perspectives, (Do et al., 2022) advocate for a sharding-based blockchain as a strategic solution to enhance blockchain effectiveness within large-scale networks. The proposal involves partitioning the network into distinct shards, enabling parallel transaction processing and reducing the quorum size for quicker consensus. Addressing uniformity challenges in existing sharding-based networks, the authors propose a system with heterogeneous consensus algorithms, showcasing substantial improvements in experimental results.

Furthermore, (Milo et al., 2022) contribute three insightful case studies, examining vulnerabilities within smart contracts, including the Dexter1 token swap, iToken, and Brave's BAT token ICO. Demonstrating the potential efficacy of property-based testing, they utilize ConCert, an executable model for smart contracts. This amalgamation of formal verification and property-based testing proves compelling for fortifying smart contract security and reliability.

In the domain of Internet of Drones (IoD), (Alsamhi et al., 2022) introduce a blockchain-based approach to manage multi-drone collaboration, addressing challenges of energy efficiency and information security. Blockchain serves as a communication tool, enhancing consensus acquisition in swarm operations, ultimately improving security, energy efficiency, and connectivity in multi-drone collaborations. This strategic integration aims to advance stability and feasibility in real-world challenges within the smart world.

To sum up, the studied approaches for testing Tezos blockchain-based applications bring notable advantages. Tezos, with its unique features and architecture, provides a robust foundation for the development and testing of decentralized applications. The use of a formal verification mechanism in Tezos, which includes a self-amending blockchain and Michelson as its smart contract language, enhances the reliability and security of applications. The ability to employ property-based testing techniques ensures a thorough evaluation of the core components, contributing to the overall software quality of the Tezos blockchain.

Our proposed solution for automated testing of smart contracts under the Tezos blockchain permits to narrow the divide between intricate blockchain development and non-technical stakeholders engaged in the smart contract ecosystem. It provides an intuitive platform designed for the testing and analysis of smart contracts, catering specifically to individuals lacking an extensive grasp of blockchain technology or programming.

4 DESCRIPTION OF OUR TESTING SOLUTION

The fundamental aim of our solution is to elucidate the intricacies of the testing procedures associated with Tezos smart contracts, rendering them accessible, comprehensible, and actionable for non-developers, including project managers, security auditors, and business stakeholders. Notably, our proposed tool, ST2A, streamlines the testing process through automation, enhancing efficiency and the user experience.

4.1 Used Technologies

Our solution seamlessly integrates various technologies to achieve its goal:

- **Electron:** At the forefront is Electron, a framework for creating native applications with web technologies like JavaScript, HTML and CSS. Electron is employed to build the desktop application interface, providing a modern and accessible user experience.
- **SmartPy CLI:** SmartPy CLI is utilized for executing the tests of the smart contracts. It serves as the bridge between the smart contract code and the application, enabling the execution of tests and generation of results in a standardized format.
- **Python:** Python scripts form the backbone of the application's logic. They are used for processing and analyzing the test results generated by the SmartPy CLI. Python's versatility and ease of use make it ideal for handling complex data processing while maintaining code readability and simplicity.

4.2 General Architecture of Our Proposed Solution

The application architecture is designed to be intuitive and efficient, consisting of the following components (see Figure 1):

- **User Interface (UI):** Developed using Electron, the UI provides a clean and straightforward layout where users can load smart contract files (written in SmartPy) and define expected test outcomes. The interface is designed to be intuitive, requiring minimal technical knowledge.
- **Test Execution and Result Processing:** Upon user input, the application utilizes SmartPy CLI to execute the smart contracts' tests. The results are then processed by custom Python scripts, which extract and analyze the data, transforming it into a more digestible format.
- **Result Comparison and Analysis:** Another set of Python scripts compares the extracted test results with user-provided expected outcomes. This comparison is crucial for verifying the smart contract's functionality against predefined expectations
- **Graphical Report Generation:** Finally, the application generates a comprehensive report in a PDF format. This report includes detailed analyses, such as success and failure rates and comparative evaluations, presented through easy-to-understand graphs and charts. This step is crucial for non-technical users, as it translates complex test results into visually understandable information.

In summary, the application stands out by providing a streamlined, non-technical interface to the powerful testing capabilities of SmartPy, thereby opening up the world of blockchain and smart contract testing to a broader audience. Its architecture smartly integrates various technologies to ensure that the complexities of smart contract testing are abstracted away, leaving users with clear insights and actionable information.

5 TESTING OF "TicTacToe" TEZOS SMART CONTRACT

In this in-depth section, we delve into the testing process of the "TicTacToe" Tezos smart contract using our proposed tool, highlighting its ability to facilitate error detection and correction, particularly for non-technical testers.

5.1 Test Setup

- **Smart Contract Tested:** "TicTacToe", a classic game implementing various interactions.
- **Tests Conducted:** Several game moves, including forbidden moves and victory validations.
- **Oracle (Expected Values):** A JSON file containing the expected results for each game action.

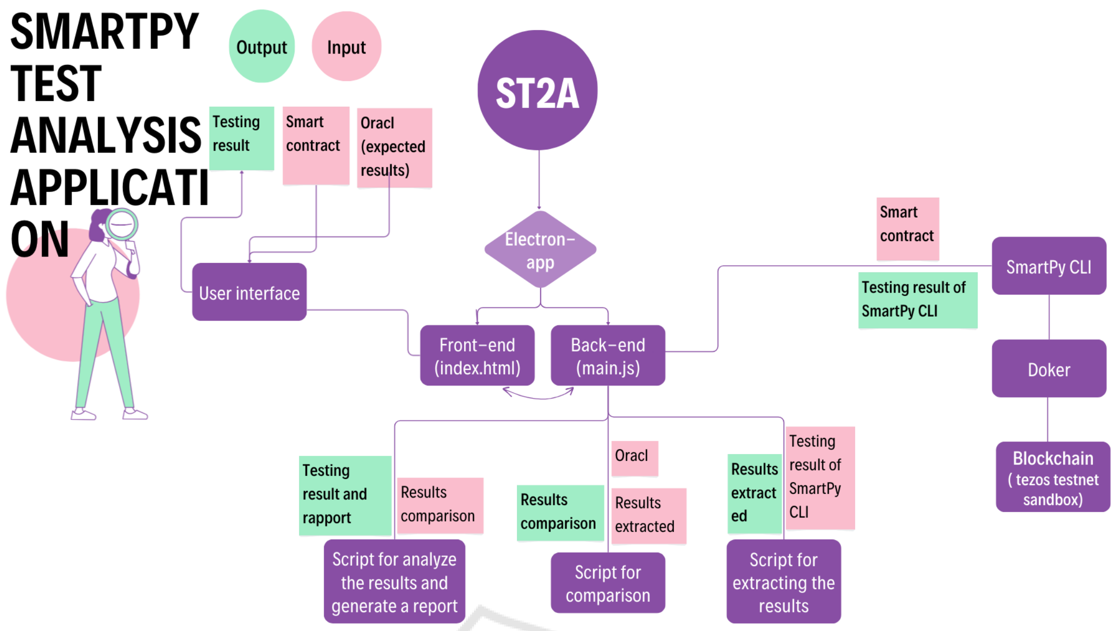

Figure 1: General Architecture of our Solution.

5.2 Execution and Extraction of Results

- **Using SmartPy CLI:** Executing the contract generates detailed logs.
- **Result Extraction:** The script extract_a_json.py processes the logs to extract key results, saved in extract.json.

5.3 Comparison of Results and Report Generation

- **Comparison:** Matching the results with expected values using compare_results.py, creating results_comparison.json.
- **PDF Report:** analyse_tests.py produces a detailed PDF report, illustrated with explanatory graphs.

5.4 Analysis of Results

- **Generated Graphs:**
  - Success/Failure Bar Chart: Visually presents successful and failed operations (see Figure 2).
  - Circular Chart: Distribution of successful and failed tests (see Figure 3).
- **Key Observations:**
  - Error Detection: The analysis revealed a divergence in the play_9 test. The expected value indicated that the second player's move should have been blocked (as the first player had already won), but the obtained value showed the move was executed. This finding is crucial as it suggests the code might be improved to include conditions that prevent such moves once the game is concluded.

Following this observation, the tester can recommend the developer to revise the game functions to better handle end-of-game logic.

5.5 Benefits for Testers

Our solution easily identified an inconsistency in the smart contract's behavior, which might have been challenging to detect without the graphs and detailed report. Thus, the tool proves extremely useful, offering a user-friendly platform for complex analysis, making smart contract testing accessible even to non-technical users.

In conclusion, this test example with "TicTacToe" demonstrates the efficiency and added value of our application in the Tezos smart contract testing process. It not only simplifies the testing procedure but also offers helpful details for the ongoing improvement of smart contracts.

5.6 Improvement Lifecycle of a Tezos Smart Contract

This section demonstrates how our application facilitates an iterative testing and improvement cycle for

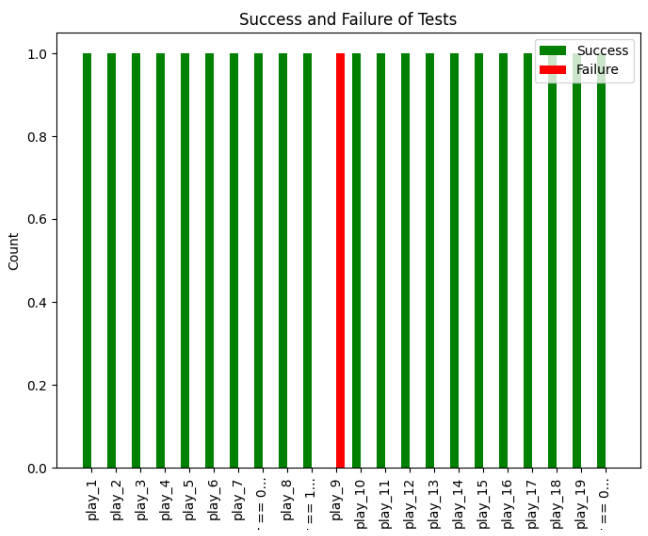

Figure 2: Success/Failure Bar Chart.

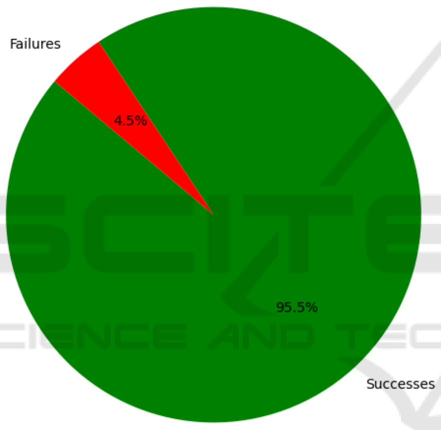

Figure 3: Success/Failure Circular Chart.

Tezos smart contracts, using the "TicTacToe" smart contract example, particularly focusing on the play_9 test case.

5.6.1 Error Identification

- In the initial testing cycle, the application revealed a discrepancy in the play_9 test.
- Identified issue: After the first player won, the second player's move should have been disallowed, yet it was executed.

5.6.2 Referral Back to Development

- Following this discovery, the tester recommends revising the code to properly handle the game's conclusion. The developer then amends the smart contract to address the identified error.

5.6.3 New Testing Cycle

- With the corrections made, the "TicTacToe" smart contract undergoes another round of testing. The objective is to ascertain whether the modification has effectively resolved the issue.

5.6.4 Post-Correction Results

- The tests are rerun using the modified version of the smart contract. The application again conducts the extraction, comparison, and report generation.

5.6.5 Analysis of Revised Results

- Test results now show a significant improvement. Particularly, the play_9 test now displays the expected behavior, validating the developer's correction.

5.6.6 100

- A circular graph generated by the application now shows a 100% success rate (see Figure 4).
- This result not only indicates the effective resolution of the issue but also our approach capability in facilitating the iterative testing process.

6 CONCLUSION & FUTURE WORKS

In this paper, we propose our ST2A solution, based on best practices and experience in applying novel

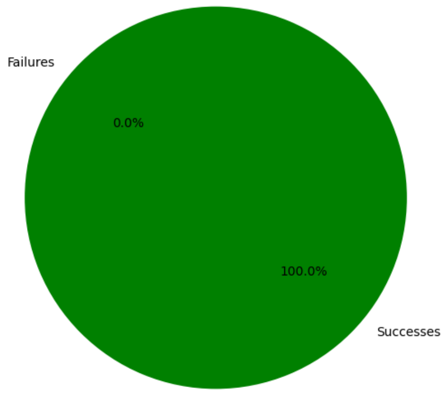

Figure 4: Success/Failure Circular Chart.

approaches, as a testing mechanism for secure Tezos smart contracts. It significantly enhances the testing process for smart contracts on the Tezos blockchain and thus can be applied to large-scale industrial projects in the context of such software architectures.

Our application automates various steps of the Tezos smart contract testing process, significantly reducing the time and effort required by testers. This automation is particularly beneficial for repetitive or complex testing scenarios. In addition, the ability to generate detailed and graphical test reports is a notable feature of our application. This approach makes test results more accessible and easier to interpret, especially for testers who may not have a technical background in blockchain technology. Moreover, by making smart contract testing more approachable for non-technical users, our application extends the capability to test and validate smart contracts on the Tezos blockchain to a wider audience and thus leads to more secure, robust, and reliable smart contract deployments.

The potential impact of our application on the smart contract testing process is substantial. As a software security verification and validation approach, it serves as a bridge between blockchain developers and less technical stakeholders, ensuring better understanding and more thorough validation of smart contracts.

As future works, we aim to automate expected oracle values using Artificial Intelligence and Machine Learning, by developing a system that can automatically generate expected values based on smart contract analysis. This feature could significantly reduce test preparation time and increase efficiency. Also, to overcome the local environment dependency limitation, integrating the application with cloud platforms could offer greater flexibility and accessibility. Besides, it would be interesting to expand the application's reach by adding support for multiple programming languages used in smart contract development, not just limited to SmartPy.

REFERENCES

(2023). Tezos blockchain. https://tezos.com/.

Alsamhi, S. H., Shvetsov, A. V., Shvetsova, S. V., Hawbani, A., Guizani, M., Alhartomi, M. A., and Ma, O. (2022). Blockchain-empowered security and energy efficiency of drone swarm consensus for environment exploration. *IEEE Transactions on Green Communications and Networking*, 7(1):328–338.

Arts, T., Svensson, H., Benac Earle, C., and Fredlund, L.-Å. (2023). Testing feature-rich blockchains. *Software: Practice and Experience*, 53(5):1144–1173.

Do, Q. H., Souihi, S., Van Van, T., Tran, H. A., and Mhadhbi, S. (2022). How tezos blockchain can meet iot? In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 2903–2908. IEEE.

Feng, R. (2023). *Decentralized Insurance: Technical Foundation of Business Models*. Springer Nature.

Gupta, V., Gupta, C., and Pereira, L. (2023). Policies for blockchain adoption in education. In *Elgar Encyclopedia of Services*. Edward Elgar Publishing Limited.

Koskinen, J. (2023). Cloud security architecture.

Milo, M., Nielsen, E. H., Annenkov, D., and Spitters, B. (2022). Finding smart contract vulnerabilities with concert's property-based testing framework. *arXiv preprint arXiv:2208.00758*.

Nishida, Y., Saito, H., Chen, R., Kawata, A., Furuse, J., Suenaga, K., and Igarashi, A. (2022). Helmholtz: A verifier for tezos smart contracts based on refinement types. *New Generation Computing*, 40(2):507–540.

Olivieri, L., Jensen, T., Negrini, L., and Spoto, F. (2023). Michelsonlisa: A static analyzer for tezos. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 80–85. IEEE.

Rahouti, M., Lyons, D., Jagatheesaperumal, S. K., and Xiong, K. (2023). A decentralized cooperative navigation approach for visual homing networks. *arXiv preprint arXiv:2310.00906*.

Sugianti, Y., Farida, E., and Athia, I. (2023). Pengaruh kompensasi, motivasi dan komitmen organisasi terhadap kinerja karyawan pada pt nuroho software consulting surabaya. *E-JRM: Elektronik Jurnal Riset Manajemen*, 12(02).

Zheng, Q. (2023). Low-cost reality capture system for high-fidelity unreal engine-based robot simulator, final year project (FYP), Nanyang Technological University, Singapore.