Acta Polytechnica CTU Proceedings **51**:109–113, 2024
https://doi.org/10.14311/APP.2024.51.0109
© 2024 The Author(s). Licensed under a CC-BY 4.0 licence
Published by the Czech Technical University in Prague

COMPLEX SYSTEM TESTING BASED ON API

Milan Sliacky*, Lukáš Kacar

Czech Technical University in Prague, Faculty of Transportation Sciences, Department of Transport Telematics, Konviktská 20, 110 00 Prague 1, Czech Republic

* corresponding author: sliacky@fd.cvut.cz

Abstract. The article describes the practical experience of testing a specific ticketing system for public transport based on blockchain technology. The testing aimed to verify the overall functionality of the complex system as well as the correct execution of the functions of its selected parts, which consist of several separately developed but closely cooperating subsystems: blockchain, clearing, and mobile application subsystem with support of bank payment system.

Communication between the different parts was implemented through APIs developed for this purpose. The availability of API documentation and the implementation on Linux OS allowed the use of sophisticated tools for testing and performing both manual and automated tests. Performed tests made it possible to detect errors in the program code of the respective components and to provide the basis for their elimination. Applied approaches and tools can be used to test similar complex information-handling systems.

Keywords: API, ticketing, public transport, blockchain, clearing, mobile application, functional test, stress test.

1. Introduction

Although the origins of blockchain technology date back to the 90s of the 20th century, it is a relatively new technology whose more massive use, unless we count cryptocurrencies, can only be observed in recent years. This technology is gradually finding applications in various areas of human life [1]. Public transport is also becoming one of these areas, specifically in passenger ticketing systems.

Blockchain has important features that distinguish it from the technologies used in this field. Blockchain is distributed, which means the data can be stored in different parts of the public transport system, i.e., on the carrier's premises and in the means of transport. It is decentralized, which has a positive impact on the security and reliability of the system in the sense of data loss [2]. Data is protected against tampering; this is a native feature of blockchain. It contains a distributed state database that is convenient for recording ticketing data (e-wallet status, loyalty program status, ticket purchase transactions, etc.). It is optionally encrypted, which means security in the sense of data protection against theft.

However, deploying blockchain in this field also brings some risks, such as the insufficient transaction processing speed caused by technological limits. At the same time, sufficient speed is essential for passenger check-in.

2. Solution used

There are a variety of blockchain implementations. The so-called industrial blockchain was used in the project, namely Hyperledger Fabric [3].

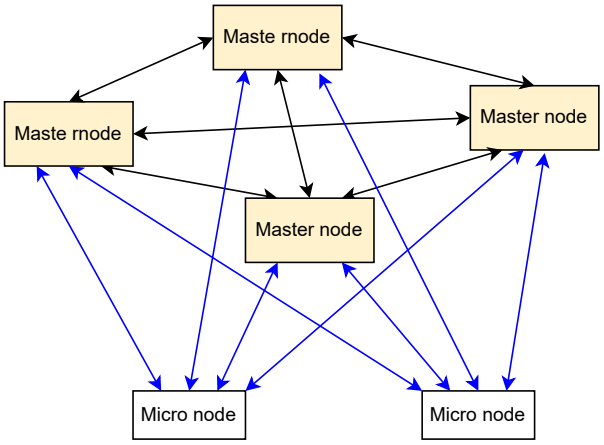

Figure 1. Principal blockchain scheme.

A common blockchain system consists of several nodes and uses several channels [3]. This project used two types of nodes: master node and micro node. The principal blockchain scheme is shown in Figure 1. The master node is fully functional and implemented as a server with a fixed location. The micro node has reduced functionality and is located in the vehicle. Each micro node processes transactions from its vehicle and can work online and offline. The blockchain payment system was designed with four channels: electronic wallet, tickets, loyalty system, and whitelist/blacklist (for compatibility with existing systems, where the blacklist contains a list of prohibited transport cards, and the whitelist includes a list of products linked to the passenger identifier). The blockchain subsystem was integrated with two other systems: the clearing

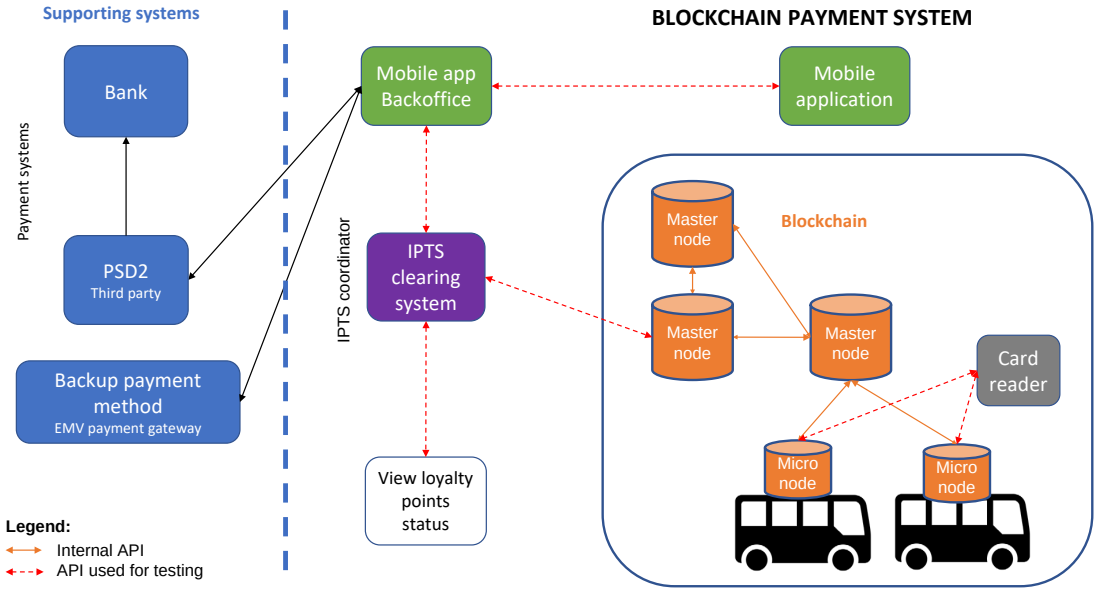

Figure 2. Blockchain payment system and supporting systems.

and mobile application subsystem. There were also supporting systems (see Figure 2) for payment.

3. System analysis

To propose a suitable testing methodology, an analysis of the blockchain payment system was first carried out to identify key properties for testing. The clearing subsystem and mobile application subsystem were also found to be important. The key features identified are as follows. The physical architecture of the whole system consists of several separate stand-alone subsystems. Information exchange (system collaboration) takes place through the APIs. All APIs are documented. Some functions (realized by separate units, like micro node and mobile app) must work correctly offline, without interaction with other subsystems. The primary function of the blockchain payment system is the registration of ticketing data (electronic wallet records, ticket records, loyalty system and whitelist/blacklist). Internal communication of the subsystems used was not subject to testing (and documentation was not provided for the testing team).

Based on the results of the analysis, research was conducted on how the industrial blockchain system works and how functional, security, and stress tests of complex systems connected via API interfaces are performed. Security tests are not the subject of this article.

4. Methodology

First, the reasons and objectives of the functional tests were assessed [4]. Functional tests verify the correctness and reliability of all declared functions of the whole system. The stress tests are similar to functional tests but under high load. Because we knew a detailed description of how the system and its parts should work, we could have used the so-called white-box testing [5].

A high system load can be obtained by quickly generating transactions at the selected system inputs, generating many offline transactions, and then switching online, generating transactions at a high load ensured in some other way or by a combination of the above methods.

Functional and stress tests were performed manually or in automated mode. The correctness of the GUI design was tested manually. Specifically, control logic and control layout were tested. Furthermore, the correctness of the response to the specified inputs was verified. The response also depends on the internal states of the system. The result was much like what you would get from an automated testing tool. Still, humans have more flexibility than automated tools, so they can follow much more complicated instructions [5]. A well-known disadvantage of manual tests is the limited number of tested combinations at the system inputs and the time-consuming nature of the tests. There is also a higher probability of random error caused by the human factor.

Automated tests allow the detection of random errors (they occur for certain combinations of inputs and internal states only). They allow testing many combinations of external and internal system states. The identified advantage of the automated testing implementation is the existence of APIs and the availability of their documentation. The following APIs were used for testing purposes: to card reader (CR API), to clearing (CLR API), to blockchain node (BLO-API), to blockchain gateway (GW-API), and back office of mobile app (DpAPI). All existing inter-

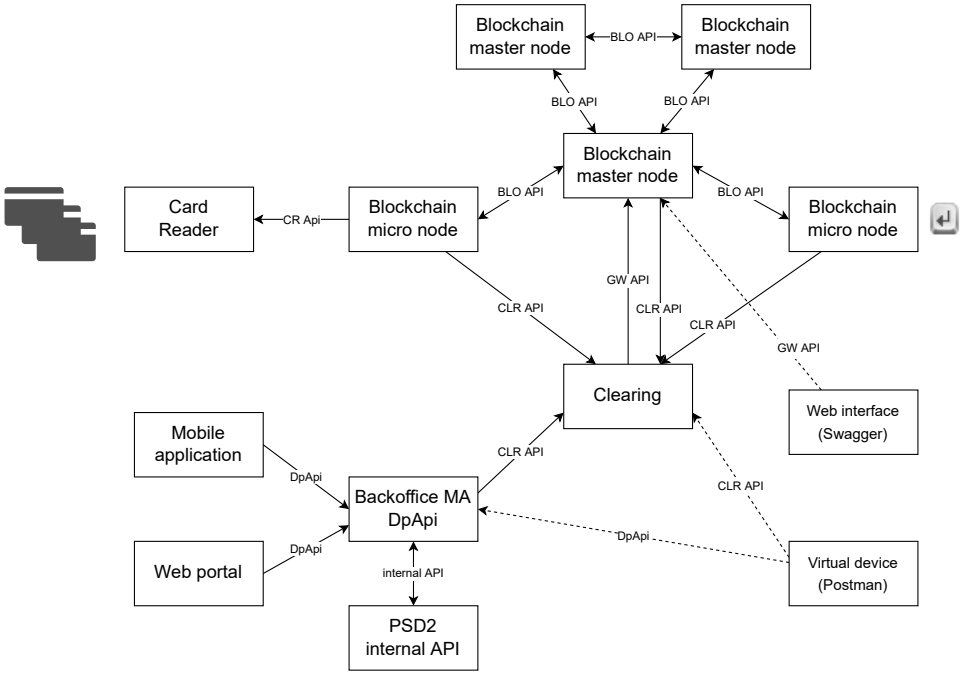

FIGURE 3. APIs available for functional and stress tests.

faces are shown in Figure 3. The blockchain native API (BLO API) is used for communication between individual nodes and was not used for testing.

The following freely available SW tools were used to implement manual and automated tests: Postman (mainly for manual API testing), Notepad++ (a useful tool for analyzing data records), and Swagger (if it has been implemented within the API).

Within the physical architecture, the micro node was implemented in two versions: industrial PC with Debian operating system and Raspberry Pi 4 microcomputer with Raspbian operating system (Debian clone). The Linux-based operating system enabled specific tools to run automated tests, such as Python programming language (scripts with loops), bash scripts, and Cron job scheduler for scheduled and/or periodic script execution.

5. Implementation of tests

The tests were conducted in a laboratory transport ticketing environment according to the existing methodology [6]. To verify the correct execution of the tested functions, it was first necessary to determine the use cases or tasks the system should perform and which should be tested. Within each use case, the processes taking place on the system were described in detail. Tested tasks are:

1) purchase a ticket in the mobile app,

2a) purchase a ticket via a transport card,

2b) top-up virtual e-wallet in the mobile app, and

3) draw loyalty points.

Based on the process analysis of the defined use cases, the following functions and features were validated. On the clearing subsystem, it was the correctness of generated/recorded values of transactions on relevant equipment (micro nodes, mobile app). On the blockchain subsystem, they were the loyalty program history, e-wallet transactions history, and transaction record of the ticket purchase. On HMI at all relevant parts of the system under test, it was the correctness of displayed values.

Quality testing can only be done for automated tests where a sufficient number of retrieved values are available. Such tests were successfully carried out for automated ticket purchases on both types of micro nodes, as well as automated e-valet recharges via clearing API. The following parameters were evaluated: transaction success rate (TSR) [7]

$$\mathrm{TSR} = \frac{\text{total number of successful transactions}}{\text{total number of attempted transactions}} \cdot 100, \quad (1)$$

and average, minimum, and maximum test execution time, median, and variance. Average test execution time is, with some simplification, equal to an average transaction duration or an average latency [8].

Each use case was tested separately. The execution of the task means the communication of selected parts of the system through the appropriate APIs. For example, purchasing a ticket in the mobile app (MA) means the communication of the mobile application with its back office, PSD2, clearing, and blockchain subsystem using the following APIs: DpAPI, Internal API, CLR API, GW API, and BLO API (see Figure 3). For incorrect test results, finding the place of the error in the system (typically an error in the API design) was necessary.

6. Test results

The testing was carried out according to the design, and many errors were revealed in most of the tested subsystems (blockchain payment system, clearing, and mobile app subsystem) in the form of a pilot plant. Next, we list the main errors found during functional testing.

When testing task 1 – purchase a ticket in the mobile app, a timestamp error in bonus program history recorded in the blockchain subsystem was detected. Furthermore, missing information was found in the sent records (transport card ID and transaction ID missing), which made it impossible to trace these transactions in the blockchain database.

In task 2a – purchase a ticket via a transport card, a similar timestamp error in bonus program history was detected. Furthermore, randomly generated duplicate transactions were detected in both the clearing and blockchain subsystems. These duplicate transactions occurred very rarely – an average of 1 wrong transaction in 1888 correct ones. Manual tests probably wouldn't have caught this bug.

In task 2b – recharge the virtual e-wallet in the mobile app, wrong timestamp entries in the e-wallet history recorded in the blockchain subsystem were detected. Furthermore, wrong values in the recharge transaction records (balance and previous state of the e-wallet) were detected in the clearing subsystem records.

In task 3 – drawing loyalty points, the transaction records in the blockchain loyalty program database had the wrong timestamp – the same as mentioned above.

Automated tests made it possible to evaluate a qualitative point of view. In task 2a – purchase a ticket via script simulated transport card use, the automated tests were performed for micro node 1 resp. 2 in two stages of system development. The first tests showed the transaction success rate equal to 80.2 % resp. 80.0 % with the average transaction duration (latency) equal to 4.82 s resp. 3.67 s. The final tests indicated the transaction success rate equal to 99,73 % resp. 99.97 % with an average transaction duration of 1.28 s resp. 1.76 s. During the integration work of the system using the results of testing, there was a significant improvement in the system parameters. Compared to [8], there is still a possibility of improving efficiency.

From the stress tests conducted, it appears that if the blockchain system is overloaded, the transactions on the micro node will go offline, which is a desirable feature. The number of erroneous transactions generated on the side of the micro node and average transaction duration (latency) increased with a higher load on the blockchain system.

7. Test results settlement

All the findings and shortcomings were incorporated into the corrections and modifications of the system under development, and the system was optimized. The tests demonstrated load resistance, fault tolerance, and practical operational sustainability. The technical and substantive merits of the project were verified in a semi-operational manner, and the operation was flawless within the specified test period. As a result of the project, a system has been built and is ready for deployment in routine operation. It is a system design that will enable mass production of similar systems for use in transport.

8. Conclusions

Testing complex systems consisting of several subsystems connected through interfaces using APIs entails the need to use a whole range of different types of testing, especially in the case of functional and stress tests. Manual tests are insufficient to verify the system before commissioning, as they do not allow capturing erroneous transactions with a low frequency of occurrence (less than one error transaction per tens to hundreds of correct transactions). Automated tests providing a large number of recorded values are very useful. Test automation can be done in different ways. It is necessary to use the options provided by the tested system flexibly.

Since both types of micro nodes ran on a Linux OS, the advantages of the Linux operating system, such as scripting languages or the Cron job scheduler, were used in the testing design to generate transactions. Another way to automatically generate many specific transactions was using APIs. Thanks to the documentation provided, this option could be used. Various specialized tools allow you to generate transactions directly through the API.

In addition to capturing random errors, another advantage of automated tests is obtaining enough data from which it is possible to calculate performance characteristics. Finally, we must also mention Notepad++, which has proven to be useful when analyzing large logs (with thousands of rows).

The described approach is suitable for testing other complex systems based on computer technology, e.g., vehicle simulators, where several independent subsystems cooperate, and communication between them is realized via API. All relevant APIs must be documented. An operational system based on Linux is an advantage but not a mandatory prerequisite.

Acknowledgements

All the activities described above were carried out as part of the R&D project, ID: CZ.01.1.02/0.0/0.0/20_321/0024952. The project was supported by the operational program OPPIK, managed by the Ministry of Industry and Trade of the Czech Republic. The main research companies of the project were Aemis, s.r.o. and BUSINESS Systems a.s. Czech Technical University in Prague, Faculty of Transportation Sciences, through its Transport Ticketing and Information Systems Laboratory, carried out the testing activity as a subcontract within the implementation of the project.

References

[1] M. del Castillo. Blockchain 50: billion dollar babies, 2019. [2024-02-11]. https://www.forbes.com/sites/michaeldelcastillo/2019/04/16/blockchain-50-billion-dollar-babies/?sh=10d4292557cc

[2] S. Roy, M. Ashaduzzaman, M. Hassan, A. R. Chowdhury. BlockChain for IoT security and management: Current prospects, challenges and future directions. In *2018 5th International Conference on Networking, Systems and Security (NSysS)*, pp. 1–9. 2018. https://doi.org/10.1109/NSysS.2018.8631365

[3] Linux Foundation. Blockchain network. [2024-02-14]. https://hyperledger-fabric.readthedocs.io/en/release-1.4/network/network.html

[4] R. Stasonis. How to implement functional test in an automated environment, DDM Consulting Services Inc, USA, 1999. [2024-02-12]. https://www.ddmconsulting.com/Design_Guides/functest.pdf

[5] R. Stephens. Chapter 13 – Testing. In *Beginning Software Engineering*, pp. 327–358. 2022. ISBN 978-1-119-90170-9.

[6] M. Svítek, J. Borka, M. Sliacky, et al. The methodology for verifying the interoperability of fare collection and information systems in public transport, CTU in Prague, Prague, 2015. [2024-02-12]. http://ois.fd.cvut.cz/dokumenty/metodika_overovani_interoperability_ois.pdf

[7] S. Mercan, E. Erdin, K. Akkaya. Improving transaction success rate in cryptocurrency payment channel networks. *Computer Communications* **166**:196–207, 2021. https://doi.org/10.1016/j.comcom.2020.12.009

[8] E. Androulaki, A. Barger, V. Bortnikov, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15. Association for Computing Machinery, New York, NY, USA, 2018. https://doi.org/10.1145/3190508.3190538