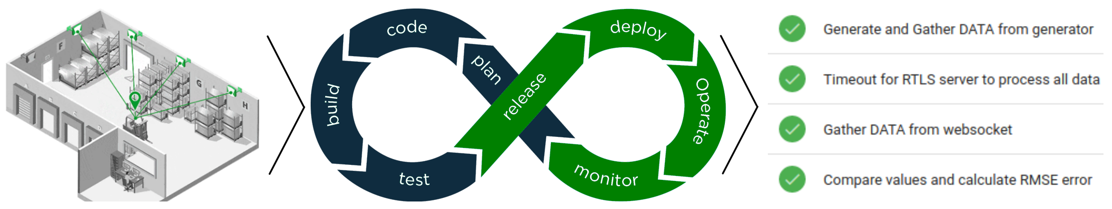
Excel@FIT 2020
http://excel.fit.vutbr.cz

Infrastructure for Testing and Deployment of the Real-Time Localization Platform

Michal Ormoš*


code
plan
build
release
test
deploy
Operate
monitor
Generate and Gather DATA from generator
Timeout for RTLS server to process all data
Gather DATA from websocket
Compare values and calculate RMSE error

Abstract
Fast development and deployment of the software are the new phenomena of the era. It is not different in the field of real-time localization systems. In our global world where the global positioning system (GPS) is the everyday utility, there is a necessity of localizing under the roof where the GPS cannot access. Here come the local position systems based on Ultra Wideband, which bring the ultimate precision. This work solves the problem of fast delivery of the software responsible for the real-time localization systems. It produces a case study on how to develop, test, and deploy this system in the continuous integration and delivery environment with the help of DevOps principles. This requires introducing the new techniques and methods for how to validate and test the precision of these systems. With these improvements, we can deliver this type of software faster by reducing the time needed for testing and validate software as it is developed. Also, we can guarantee and demonstrate it's quality across versions easier.

Keywords: RTLS systems — UWB — CI/CD — Indoor Localization

Supplementary Material: *N/A*

*xormos00@stud.fit.vutbr.cz / mi.ormos@gmail.com, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

In recent years I have been focusing my attention on indoor localization systems and products for solving the problem of indoor asset localization. I am part of the team developing the real-time localization systems (RTLS), and I was not always satisfied with how the software part of this system was tested and delivered. Recently, I started to think about how to improve this process with conventional testing methods and also anew developed methods only for this type of software. With the help of the crucial test-suite for the RTLS system, we can deliver the new features faster and more securely than before.

A real-time location system, as the name characterizes are systems requiring a zero-delay response to their inputs. RTLS system is a system based on familiar software architecture with a database for storing the data, back-end for serving the data, and front-end for presenting the data. However, in the heart of every RTLS system is location algorithm. At input of this algorithm are thousands of data streams from location hardware. Streams of this data, in raw Ultra Wideband (UWB) patterns, goes through the entire system. Data are parsed, calculated, optimized and shown in an understandable format. This data can be used for example in logistic applications, navigation and asset tracking. Guarantee that this data will be calculated as fast as possible in the correct way is crucial. As this system is growing and optimizing, we have to ensure that this will be preserved across the new releases.

Indoor localization based on UWB is quite a new concept in early development and deployment in the industry [1, 2, 3]. However, testing and validating the software products is old as software development itself. In half of the problems, we can easily apply conventional tools with little adjustments. However, the other half requires new methods and the dose of creativity. The ideas of this infrastructure come from the DevOps principles [4, 5]. It is more the right philosophy than the right tool. It is combination of the word's development and operations. DevOps assimilates development and operations teams to improve the collaboration process. A DevOps Engineer will work with IT developers to facilitate better coordination among operations, development, and testing functions by automating the integration and deployment processes. Continuous integration and continuous delivery (CI/CD) is often referred to as pillars of successful DevOps [6, 5]. To establish and optimize the CI/CD development model and receive the benefits, companies need to build an active pipeline to automate their build, integration, and testing processes.

The second half of the problem is coping with the difficulty of how to abstract this system from their hardware components. As we want to test fast, remotely, and in a stable environment, we have to abstract hardware in a software manner. All RTLS systems are dependent on devices which are transmitting the signal as RTLS Tags or merely mobile devices as well as devices receiving the signal, called RTLS Anchors [1, 7]. These devices record the signal representation of the localization data where the RTLS software then transforms this data to Cartesian coordinates system understandable by everybody. The unique part of this work will be to abstract this hardware credible so we can test the software without it as we have it.

The outcome of this work is test suite, which accelerates the delivery of the new version of the RTLS system and ensure safer and more secure development with confidence in the delivery process.

2. Background

2.1 Indoor Localization

A local positioning system (LPS) is a navigation system that provides location information anywhere within the coverage of the network in all conditions. If there is an unobstructed line of sight to three or more signaling devices of which the exact position on the place is known. A particular type of LPS is the real-time locating system (RTLS), which also allows real-time tracking of an object or a person in an enclosed area such as a buildings and factories [8, 9].

Ultra Wideband (UWB or ultraband) is any radio technology that has bandwidth exceeding 500 MHz or 20 percent of the arithmetic center frequency, whichever is lower. UWB is a carrierless communication scheme. The early applications of UWB technology were primarily related to a radar. A UWB-based locating system is very much like any other RTLS except that it uses UWB signals [4].

2.2 CI/CD Pipeline

A CI/CD pipeline helps you automate steps in your software delivery process, such as initiating code builds, running automated tests and deploying to a staging or production environment. Automated pipelines eliminate manual errors, provide standardized development feedback loops and enable fast product iterations [10].

2.3 Implementation

Software testing is the most frequently used method for verifying and validating the quality of the software. Testing is the method of executing a program or system to detect faults. It is a significant activity of the software development life cycle. It helps in developing the confidence of a developer that a program does what it is intended to do.

3. Implementation

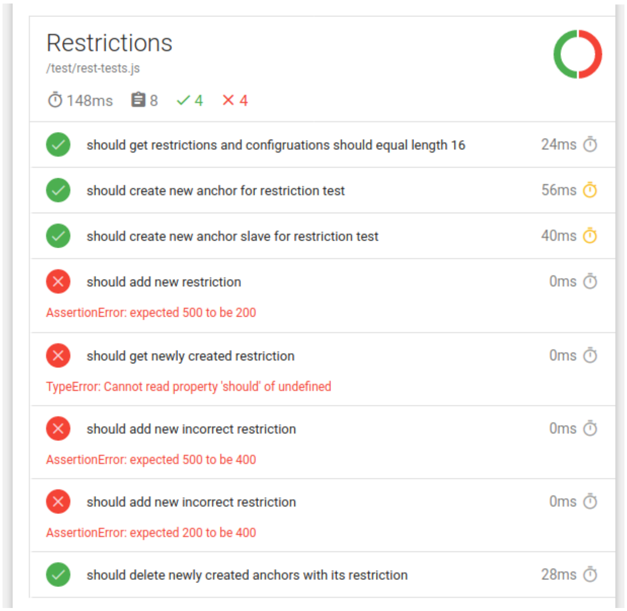
RTLS system consists of hardware and software parts. In this test-suite, we only aim for the software parts and omit the hardware parts. But as the software is dependent on the hardware inputs, we will create the hardware abstraction for software testing. This will make testing and development faster and more easygoing.

RTLS system software consist of RTLS Manager application which is communicating with the hardware and is parsing the input data. RTLS Server application for calculating the positions and RTLS Sensmap for displaying the localization data. This all is back-up with the database.

The next section deals with the individual parts of the test-suite as we will call our final pipeline containing all test cases.

3.1 REST Tests

Many third party applications are connecting to the RTLS system and are fetching the data. The most convenient method to fetch this data is WebSockets and Open REST API. In this section, we aim for the REST API. This API consists of many requests for getting the data from the database. The test-suite aims for all these calls (GET, POST, PUT and DELETE). It


Restrictions
/test/rest-tests.js
⏱ 148ms 8 ✓ 4 ✕ 4

```
✓ should get restrictions and configurations should equal length 16   24ms
✓ should create new anchor for restriction test                       56ms
✓ should create new anchor slave for restriction test                 40ms
✕ should add new restriction                                          0ms
AssertionError: expected 500 to be 200
✕ should get newly created restriction                                0ms
TypeError: Cannot read property 'should' of undefined
✕ should add new incorrect restriction                                0ms
AssertionError: expected 500 to be 400
✕ should add new incorrect restriction                                0ms
AssertionError: expected 200 to be 400
✓ should delete newly created anchors with its restriction            28ms
```

Figure 1. Example of an invalid test from the REST Tests section of our test-suite pipeline. Individual test cases are self-explaining and conditions are described.

tests the positive and negative results of these requests. By these tests, we can be sure the RTLS system always responds to the API call as it was designed to. The part of testing WebSockets will be mentioned in the next sections.

3.2 Syntax Checkers

RTLS systems usually have many contributors within the team. DevOps principles emphasize that the same tools and the same principles have to be used across the team to guarantee the quality and validity of the software. By this principle, we introduce the linters. Lint, or a linter, is a tool that analyzes source code to flag programming errors, bugs, stylistic errors and suspicious constructs. The team settles on the shared rules for writing the source code and these rules will be forced in the pipeline. If the developer do not obey this rules, the CI/CD pipeline will reject his commit.

3.3 Unit Tests

Unit tests are the first level of the testing process. They are validating the individual parts of every module. The unit test is written in a white box method, where the developer writes the tests based on information gathered from the source code. From this point the test-suite will be in a black-box method where the tests are written only with the specification of what the software should do and without the information from the source code.

3.4 Performance Tests

An essential part of our test-suite will also be to find out where are the performance limits of the software implementation depending on the hardware organizations. The limitation of hardware parts can be easily calculated by their design and theoretical limits of the UWB. Nevertheless, as always, the software is usually built together with many third-party applications and running on different operating system distributions, so a real performance test under the load has to be performed. Our performance tests will be included within the REST Tests as well as our Report generator tests.

3.5 pcap Player

This is the first unique test case built for the RTLS system. As RTLS systems are already built and deployed in the installations around the globe, we can use this in our benefit. In the first way of application testing we take the network recordings of the real application. Then we eliminate the packets that are not connected to the localization itself. After that, we replay this recording to the RTLS system.

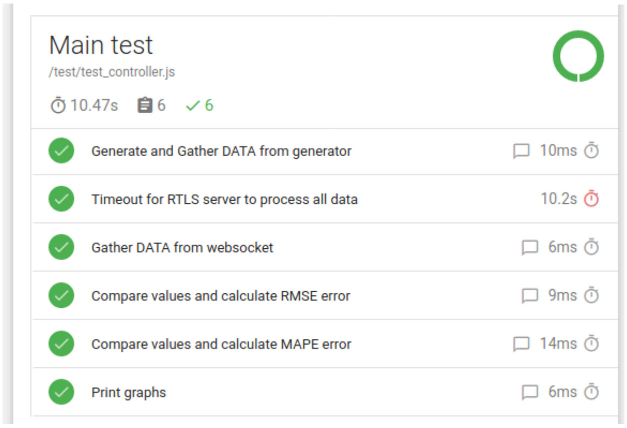
But before that, we need to take the original system database and settings, and import them to the system where we are replaying. Thereby we can replay the same scenario to the system again and again. By this method we can ensure that the system works the same across the distribution and we can easily tune the system remotely.

3.6 Report Generator

In the pcap Player, we were limited to the same payload which we could send to the RTLS system. In this part, we create the generator of our unique payload. This will give us the benefit to create our own scenarios. Hereafter we mention RLTS Tags as tags and RTLS Anchors as anchors.

Tags are sending the signal to the environment and Anchors are receiving them. Tags are dynamic devices in the movement and Anchors are static devices with a predefined position. The most important part of Anchors is the synchronization between them. The RTLS system is receiving the RTLS Blink signal from Anchors about the tag signal time of flight from Tag to the Anchor, and also the RTLS Sync signal about the synchronization between the Anchors. By learning how this Blinks and Syncs are created and successfully generating them we can create authentic RTLS payload generator.

To have full control of creating UWB payload by the software we can now create random scenarios that suit our purpose. Moreover, most localization errors originate in synchronization inaccuracies. By that we


Main test
/test/test_controller.js
10.47s 6 ✓6

✓	Generate and Gather DATA from generator	10ms
✓	Timeout for RTLS server to process all data	10.2s
✓	Gather DATA from websocket	6ms
✓	Compare values and calculate RMSE error	9ms
✓	Compare values and calculate MAPE error	14ms
✓	Print graphs	6ms

Figure 2. Example of a valid report generator test in our test-suite. Passing on all items, which are self-explaining.

mean that we can put intended errors to our synchronization data on purpose. Thereby we can observe how the system and his filters cope with the errors and use some statistical methods as RMSE or MAPE to fully validate this scenarios.

3.7 CI/CD integration

To make this pipeline work, the final step is to put it to the system code repository where the project is developed and ensure this pipeline will run on every significant change of the code in the repository system to preserve system quality and validity. We decided to use the GitLab repository system, where the codes are already stored. GitLab offers CI/CD built in every repository based on GitLab runners, an open-source project that is used to run your jobs and send the results back to GitLab. It allows you to set different scenarios when the test should run, as for example ours at every commit or every midnight.

4. Evaluation

In this short evaluation, we introduce the final output of the pipeline and introduce a few methods of how can we validate it. In REST Tests, Unit Tests and Syntax Checker we are validating results pretty straightforwardly. We will specify limits and values when the test passes or fails.

This was already shown in Figure 1 and Figure 2. We set conditions in which individual tests success or fail, and the output of our test pipeline results in these conditions passing or failing based on data they received.

But this is not so straightforward in pcap Player and Report Generator. In Figure 3 we can see the output of the report generator as we plot the data to the plan. The orange line representing the data generated by the report generator. This data are transformed into data similar to UWB traffic, as Anchors would generate them and passed to the RTLS software on its input. The green line represents this UWB traffic passed through RTLS system and positions are calculated. By this simple demonstration, we can assume our report generator is working as we design to. This is suitable for manual validation but unacceptable by pipeline.

One way we can address this is to use statistical methods as Root-mean-square deviation (RMSE equation 1) and Mean absolute percentage error (MAPE equation 2) and calculate the deviation of points generated compared to points gathered as shown in the Figure 4. Variable $\overline{y}$ in both equations represents predicted values, variable y, observed value. Variable n in both equations means number of observations.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\overline{y}_i - y_i)^2} \tag{1}$$

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{\overline{y}_i - y_i}{y_i}\right| \tag{2}$$

Calculated MAPE and RMSE of our evaluation case can be seen in Table 1. Here we can see deviation of x-axis ans y-axis respectively. Deviation is calculated between the every pair of points as shown in Figure 4. Orange points are from report generator and greens points are from RTLS system localization output.

	Values	
Method	x-axis	y-axis
MAPE	2.57%	1.05%
RMSE	6.12801	1.41575

Table 1. Localization accuracy.

By this approach, we can determine the boundaries of deviation where the localization is acceptable and when unacceptable.

5. Conclusions

We introduce an indoor localization system based on the UWB. We create test-suite based on CI/CD and DevOps principles. This test-suite is specially designed for the RTLS systems. It consists of commercially used tools as well as our proprietary tools designed and created for testing RTLS systems without the necessity of real hardware components.

By applying this pipeline, we reduce delivery time of the new RTLS system version. As this systems are delicate and their real-time principles are crucial for

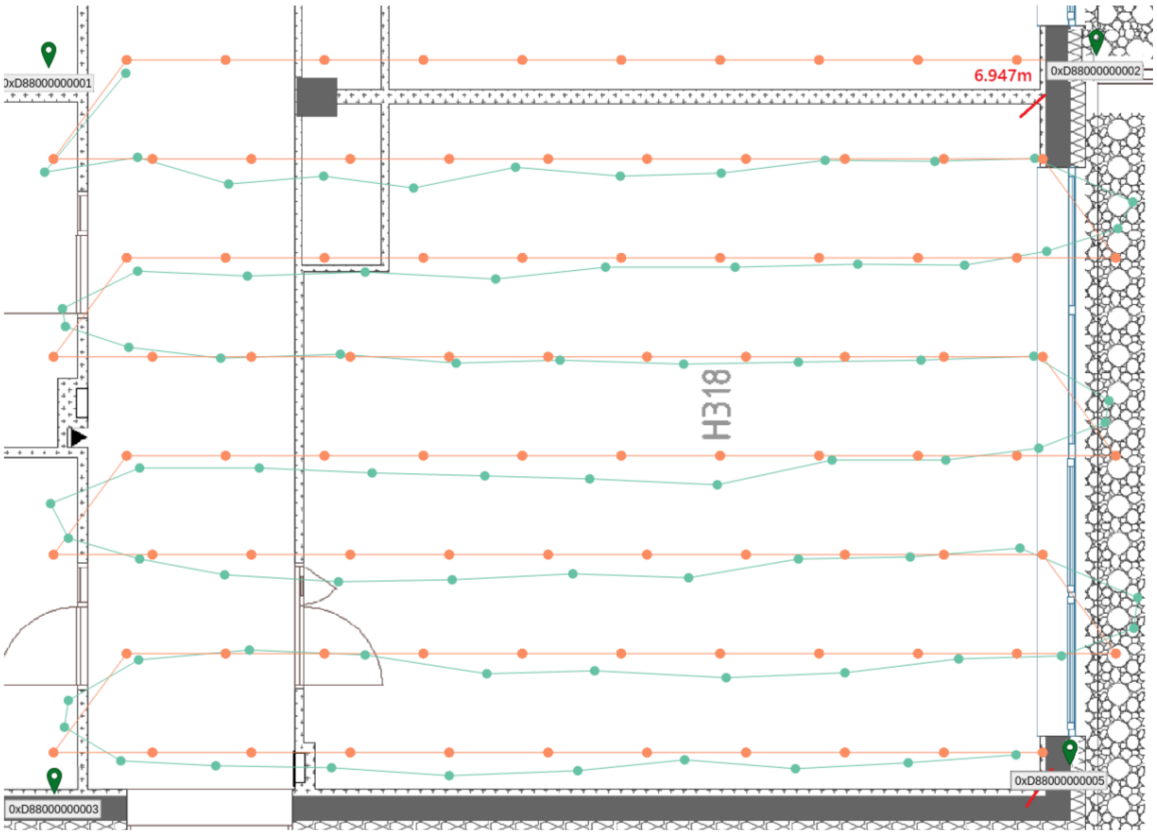


Figure 3. The demonstrative representation of the report generator. The original trajectory from the report generator (orange) with output trajectory from RTLS system (green). Localization is never a straight line due to many approximation filters algorithms of RTLS system use. The localization movement starts from the top right corner and going from side to side until finishing in the bottom right corner.

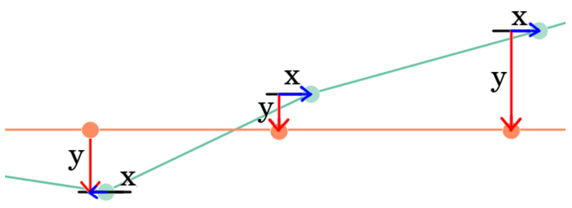


Figure 4. Example of how we use RMSE and MAPE for automatic pipeline validation for deviation in both axis of localization.

logistic and business applications, we can now deliver this software more securely and with validation that everything works as it should.

This paper is a case study of RTLS systems and describes how to deliver them quickly and safely. It analyzes this new rising technology and necessity to introduce the way it can develop and deliver with confidence.

This pipeline can be easily adjusted for different RTLS systems in the future. But for this particular UWB technology it could for the future include of more statistical metrics for validation, as well as front-end test cases for even faster CI/CD process.

Acknowledgements

I would like to thank my supervisor Ing. Vladimír Veselý Ph.D., who helped me by leading this thesis through the academic environment. I also want to express my thanks to employees of Sewio Networks s.r.o, who helped me with their knowledge about the RTLS system I worked on, and allow me to create a thesis on that exciting and rising topic.

References

[1] Len Bass. An evaluation of indoor location determination technologies. *IEEE Software*, 35(1):8–10, 2018.

[2] Fouzia Boukour Elbahhar and Atika Rivenq. *New Approach of Indoor and Outdoor Localization Systems*. InTech, 1 edition, 2012.

[3] N. Oster F. Saglietti and F. Pinte. White and grey-box verification and validation approaches for safety- and security-critical software systems. *Information Security Technical Report*, 13:10–16, 02 2008.

[4] L. Bass. The software architect and devops. *IEEE Software*, 35(1):8–10, January 2018.

[5] Joost Evertse. *Mastering GitLab 12: Implement DevOps culture and repository management solutions*. Packt Publishing, 1 edition, 2019.

[6] Adam Debbiche, Mikael Dienér, and Richard Berntsson Svensson. Challenges when adopting continuous integration:: A case study. volume 8892, pages 17–32. Springer, Cham, 2014.

[7] Mengda Wang, Bing Xue, Wei Wang, and Junjie Yang. The design of multi-user indoor uwb localization system. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, volume 2017-, pages 322–326. IEEE, 2017.

[8] Johan Hjelm Krzysztof W. Kolodziej. *Local Positioning Systems LBS Applications and Services*. Taylor Francis Group, LLC, 1 edition, 2006.

[9] Sewio public documentation. online, 2019.

[10] Manish Virmani. Understanding devops bridging the gap from continuous integration to continuous delivery. pages 78–82, 2015.