Introduction to Lean Canvas Transformation Models and Metrics in Software Testing

Padmaraj Nidagundi[1], Leonids Novickis[2]

[1] *IITMinds, India,* [2] *Riga Technical University, Latvia*

Abstract **– Software plays a key role nowadays in all fields, from simple up to cutting-edge technologies and most of technology devices now work on software. Software development verification and validation have become very important to produce the high quality software according to business stakeholder requirements. Different software development methodologies have given a new dimension for software testing. In traditional waterfall software development software testing has approached the end point and begins with resource planning, a test plan is designed and test criteria are defined for acceptance testing. In this process most of test plan is well documented and it leads towards the time-consuming processes. For the modern software development methodology such as agile where long test processes and documentations are not followed strictly due to small iteration of software development and testing, lean canvas transformation models can be a solution. This paper provides a new dimension to find out the possibilities of adopting the lean transformation models and metrics in the software test plan to simplify the test process for further use of these test metrics on canvas.**

Keywords **– Lean canvas, software testing, software validation, software verification, test process.**

I. Introduction

Now the growing technology makes human depend in day-to-day life more on software embedded devices. Software itself plays a key role in human life. The software development has also changed from the past decade and new development methodologies are introduced by new fast software development and testing concepts such as agile. On the one hand, fast development of software is going on and, on the other hand, fast validating and verification parallel supporting built high quality error free software in a fast way according to the requirements.

In a software development company the software testing process gives more power to build high quality software. In recent years software testing has gained more value proposition that can directly impact the software building and delivering process. The lean canvas transformation models may bring a new dimension to software testing overcoming many obstacles such as test plan, long test documentation, test resource planning to software release.

The word “lean” comes from lean manufacturing and shows the elimination of unnecessary waste “muda” from the process and brings the value proposition of the overall process. The lean concept is re-used by Ash Maurya for the lean canvas for the business [1]. Till nowadays lean canvas was used as a business mode and for defining a business plan, a problem, solutions, key metrics and possible advantages.

The lean process concept is focused on the removing of the waste and improving the efficiency of the development process. This can have a direct impact on speeding up the software development process and reducing the development time as well as the cost. Meanwhile a company can develop high quality software in the correctly estimated timeframe. Many studies have shown that lean main advantage is delivering the product earlier than the planned time. It means that the software developer team develops and delivers more product functionalities in short time and the testing team can verify and validate them. This can also influence the project finance and directly customers. The adoption of the lean process concept in software development improves the decision making in the team and keeps the team motivated all the time. This way it can affect the quality of the final product of the customer.

A. Scientific Novelty of Paper

Lean canvas is created to evaluate the business model. It is one white board with several segmentations that mean a lightweight one-page document that shows the product creation to evaluate marketing fit.

Our core contribution is to highlight the identified test metrics and possibilities of adoption of the use of lean canvas transformation models in the software test process.

Specifically:

- To identify the entry criteria, exit criteria for the software product testing.
  - In waterfall model where software testing process at the end and long test documentation are time consuming.
  - In agile where software development cycle is very short and software testing needs to be done in time and there is very less time for lightweight test documentation creation.
  - In software test documentation, where it is very difficult to cover all test items.
  - In test planning where before software testing starts a test plan must be ready.
- To evaluate possibilities of adoption lean canvas transformation models in software testing.
- To find the way to generate different test metrics, which affect the software testing.

B. *Structure of Paper*

The current paper is divided into the three main sections and conclusion. Section 1 formulates the problem and Section 2 provides the brief introduction of related studies to business model canvas by using modern approach. Section 3 shows testing process and indication of lean test metrics for further utilisation for the canvas board.

II. The Problem We Tackle

We have many challenges which concern the software test planning to manually test as well as automatically till delivering tested software to end customer. In the manual testing process, a developer develops software and assumes its tester job to test it, and the company does not have any test process. A tester's responsibility is to pick up and test the item; thus, a tester is a middleman between a developer and an end customer, a key person who is directly responsible to bring the high quality software [3]. The most common challenges a tester can face:

- Complete testing of the software application:
  Is it possible? To test all possible amalgamation including manual testing as well as the automation one, there may be a combination of thousands of tests needed. With a large number of tests a tester is never able to deliver the tested software product to an end customer.

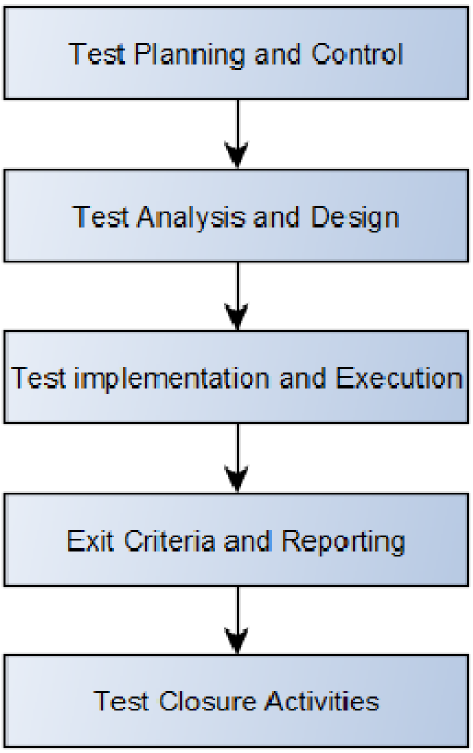
Test Planning and Control
Test Analysis and Design
Test implementation and Execution
Exit Criteria and Reporting
Test Closure Activities

Fig. 1. Block diagram for the test process overview.

- Misinterpretation of the test process at a company:
  Sometimes at a company a tester is not very much focused on the well-defined test process and sometimes a tester follows the whole well-defined process but still it does not apply to its current testing scenario. This all leads to the imperfect and unsuitable software product testing.
- Person-to-person communication with developers:
  Sometimes a big problem in a software development company is communication between the developer and the tester or the test team. The tester needs to know how to manage the relationship and keep good communication skills.
- Verifying the software in regression testing:
  When the software project has grown and become more complex, the time of regression testing becomes unrestrained. Regression testing thrusts to manage the functionality changes comparing the new with the previous developed software and reports defects.
- Absence of the proficient tester:
  It is a pure management decision to add the proficient tester or not well trained tester in the project and expect high results. This leads towards the poor tested software product to end software users.
- Testing time restriction:
  Most of time, we have noticed the boss comes and says we are releasing the software soon. Now the tester is in a hurry to complete the whole list of tasks within the time restriction. On this way the tester can focus only on the completion of the work and not on the test coverage.
- Executing the earliest test cases:
  This is pretty challenging to take the right decision for which test cases are very important and need to be executed with of priority. This is under pressure all the time.
- Comprehension of the requirements:
  In some situations the tester is directly responsible for the communication with the end customer to analyse and comprehend the requirements. In most situations the tester needs to focus very much on the listing and understanding the requirements to test the software in the right way.
- Exit criteria for software testing:
  To make a right decision to stop testing is an arduous resolution, the pre-defined exit criteria can help simplify this process. It is a very important step where all test processes get stopped and this decision is either made by the tester or the whole team together with the confident.
- One test team or a team member can be in a different project at the same time:
  It is a bit challenging to keep the track of each task. Communication gaps lead towards the failure of both or one of the projects.
- Reuse of the test document and test scripts:
  In software development the process of software application changes rapidly and it is still difficult to manage the test script and tools. Re-utilisation of the test document and test script is very much needed and it is a complex task.
- Managing the resource change:
  If an experienced tester leaves the company and an unexperienced tester is nominated in his place, the

company might face a big problem. The real challenge is to train the new tester in the specific project from the beginning and it delays the production of the well tested software.

- Test automation:
  It is always challenging what to automate and what not to and what level of automation we need with software and how it can affect the test coverage.
- Agile testing:

Agile has brought the concept of the time-boxed development, where sprints are managing the development process within a short time and a limited scope. Agile also adopted the cutting edge process such as continuous integration, where the developer checks-in the code several times a day and at the same time it is recompiled [2]. It shows that software is continuously changing. In agile development software is built in smaller development cycles and the requirements are also small to fit in the sprint. Short development cycles are focused on the fast testing of the software that builds in the sprint as well as covers the all possible tests.

Our approach: In software development process testing related problems are identified and these problems can be managed using lean canvas transformation models. Introducing the possible solutions to make the software testing process more productive different software development methodologies with lean canvas must be applied.

III. Related Research

Alexander Osterwalder introduced the business model canvas for the first time. After that it has become widely used with startup companies. The main purpose of the lean canvas is to evaluate a business model and products planned to build and market. The lean canvas validate the "business plan" and focuses on the scale of the business.

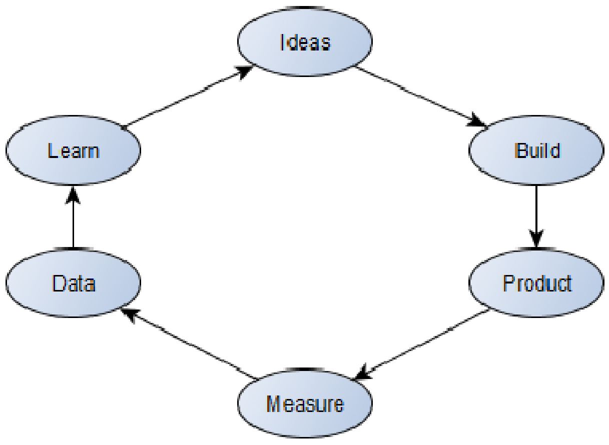

Fig. 2. Lean canvas life cycle.

Figure 2 demonstrates the full life cycle of the lean canvas. Everything starts with an idea: using this idea you want to work on it and it flows towards building the idea and taking it to the next stage of making it as a product. Once the product is ready we can measure it with different metrics. Metrics generate some useful data and we can use the data for the learning process to improve the idea. You will notice a complete loop of the canvas that helps validate the business idea.

A. Lean Canvas Meta Principles

- Make a document of your plan A
- Make an identification of the hazardous parts of your plan
- Regularly test your plan

The meta principle above helps identify the hazardous parts of your plan. There are three stages, which help to identify hazards.

First Stage: Problem/solution fit
Second Stage: Product/market fit
Third Stage: Scale

B. Lean Canvas Life Cycle

The principle from the lean startup (as shown above in Fig. 2): A) Ideas b) Build c) Product d) Measure e) Data f) Learn.

From above we can now generate the lean canvas for the business validation.

From the formulation above we can identify a similar term in software testing.

Step 1: Ideas = Software Components
Step 2: Build = Development
Step 3: Product = Testing
Step 4: Measure = Test Measure
Step 5: Data = Test Data
Step 6: Learn = Lean and Feedback

IV. Transformation of the Lean Test Life Cycle

As seen in the previous chapter about the meta principles and lean canvas life cycle, it validates the business. Now adopting the same principles and identifying and transferring similar steps to software testing we can possibly have a new lean canvas software life cycle.

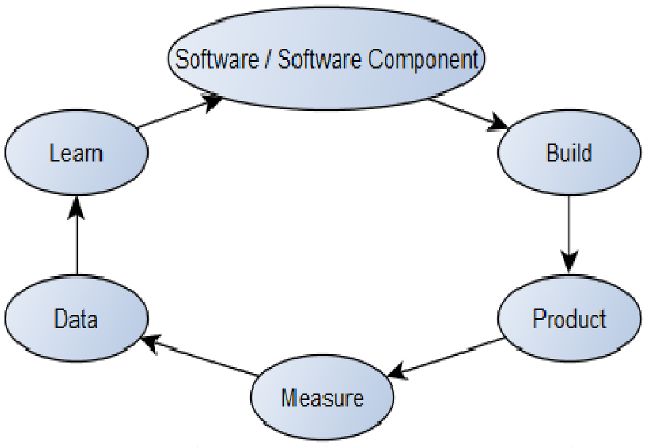

Fig. 3. Abstract of lean canvas life cycle to lean test canvas life cycle.

V. Different Software Development Life Cycles & Software Test Life Cycle Impact on Lean Test Life Cycle

A. *Different Software Development Life Cycles*

Software building is continuously changing from one decade; day to day new tools are introduced on the market to build the high quality error free software to end customers. A waterfall model develops a methodology adaptation nowadays and it slows down against agile (Scrum, DSDM, FDD, Lean, eXtreme programming, Crystal). The modern new DevOps have brought one more new dimension to software development industry.

Waterfall methodology: It is a traditional approach of the software development where first all requirements are gathered, designed, coded, tested and maintained. As we have noticed, software testing makes its software delivering process slow and increases the risk of the software failure.

Agile: In February 2001 agile manifesto for agile software development was released. The core principle of the agile software development focuses on the a) induvial and collaborative work b) building working software c) stakeholder's collaboration d) a quick response to changes [17].

DevOps – At the agile conference held in 2008 the agile infrastructure team introduced the "DevOps" [18]. It is a cultural change that focuses on the collaboration of the development, operation and QA team and makes them work together.

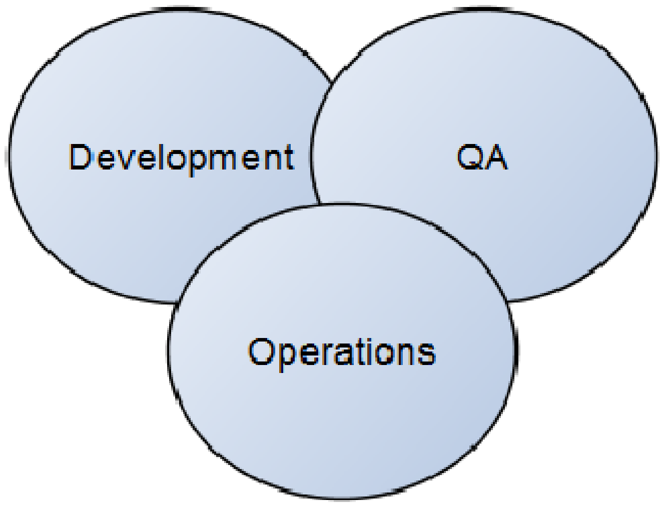

Fig. 4. DevOps work environment.

A number of tools are used in DevOps to make them more productive, the tools are set together and are called "DevOps Toolchain"

Code – Programming and code review use continues integration tools.

Build – Version control for code merging.

Test – Software testing delivers the performance.

Package – Store the code base in an artefact repository.

Release – Release automation with tools.

Configure – Configuration management tools used.

Monitor – End user experience can be monitored with complete application with different tools.

Software development and testing – There are several methodologies in practice at a company but the core challenge is to build the high quality software in a fast way and to deliver it to the end customers. The software testing also becomes very complicated with growth of different hardware and software and one must support them simultaneously. Direct impact of software testing is to meet the requirement that is defined in the software development and respond to all kind of inputs correctly and produce the correct output [10]. In traditional way, software testing is divided into two ways: white-box and black-box testing.

White-box testing is used in the unit, integration and system testing process. In white box different techniques [4] are used such as API testing, code coverage, fault injection, mutation & static testing.

Black-box testing is testing the application without knowing the application. The black box includes methods such as fuzzy testing, boundary valve analysis, use case testing, exploratory testing, etc.

Automation testing – The growth of test driven development and continuous integration tool usage increase the software test automation [7]. Test automation can speed up the testing process and help in regression testing.

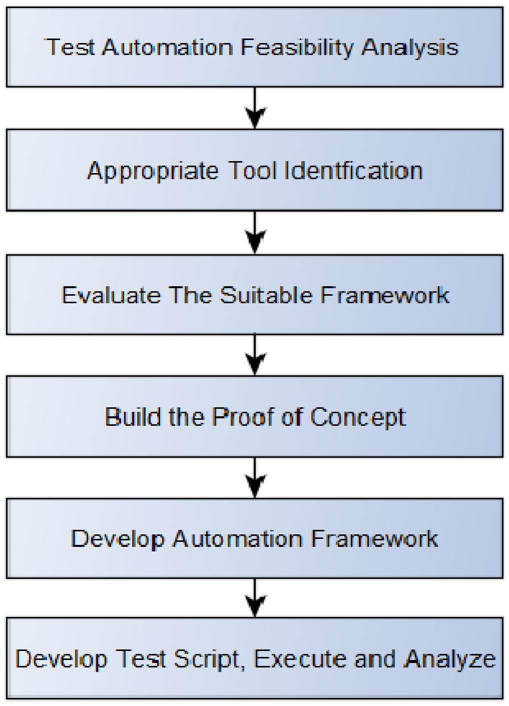
Test Automation Feasibility Analysis
Appropriate Tool Identfication
Evaluate The Suitable Framework
Build the Proof of Concept
Develop Automation Framework
Develop Test Script, Execute and Analyze

Fig. 5. Overview of the automation testing process.

As seen in Fig. 5, in the first step check we are able to automate application or not, and identify what level of automation is possible. In the next step we can select a more appropriate tool that gives a maximum benefit. In the next step we select a correct frame work that supports the tool.

In the next step we build the proof of concept (POC) for the end-to-end test case scenario evaluation. The end-to-end test case automation can prove that the software all-important functionalities can be automated.

In the next step the developing of the automation frame work is carried out in such a way that it works like an intelligent platform, which uses the test automation tools. In the final step test cases are executed by scheduling at any time without much human effort [9].

B. Different Test Life Cycles

Software testing is not a single activity or process. It consists of many other sub-activities and these all sub-processes are called an STLC.

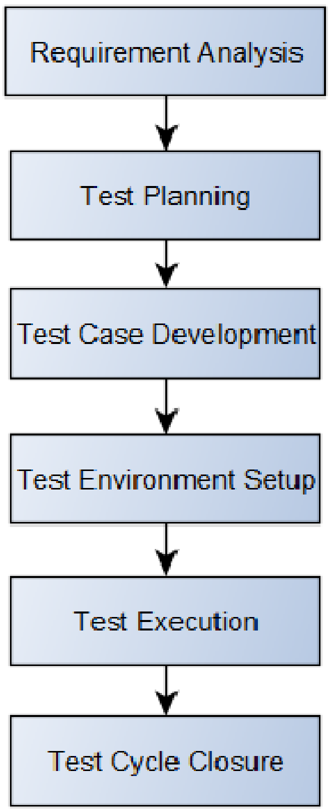

Fig. 6. STLC overview.

There are different test life cycles in use in the software industry due to different developments and test processes companies adopted. Test life cycles are called STLC – Software Testing Life Cycle. In the requirement analysis the test team focuses on the understanding of the requirements in a clear way from the client, business analyst technical leads, and system architects.

Once QA team has the requirement then tests are classified as functional, non-functional and test automation possibilities are analysed. In the test planning phase, we calculate the approximate cost and efforts estimation for the project.

In this phase we also determine the test tool selection [5], resource planning and responsibilities, which are distributed among the team members. In the next phase of the test environment setup software hardware and software conditions are determined.

At the same time, test team works on more technical topics such as architecture, hardware and software test environment. Details are focused on by the test team. In the test execution phase the test team will start working according to the test plans, execute the test cases and found software defects are reported.

In the final phase the test team members come together and discuss and analyse the testing they have done till now and review to learn lessons from the overall STLC test and identify the best practices for similar future projects.

VI. Testing Process Waste and Right Test Metrics Identification Using Lean Principles

A. Lean Principles for Identifying the Waste in Testing

Lean development principle focuses on the seven core points.

1) Elimination of waste – It is very important to investigate what kind of waste we are having in the test planning to process if it is related to documentation, regression test, test resource planning or something else. Proper test functionality, tool section, defining test criteria help eliminate the waste [16].
2) Increase learning – Educating the team on a constant basis with new changes in software can save a lot of time in testing. Software introduces new functionality and the tester gets to know at the end of such a situation which he needs to avoid. Sometimes the tester finishes his work and waits for a very long time which is also because of the lack of test management process knowledge.
3) Judge as late as possible – This situation needs to be avoided and a delayed decision costs much more in software testing, so better to adopt the right method of testing to the complete product.
4) Distribute as fast as possible – If a software item can be tested in a distributed way it brings the fast output in terms of finding possible defects very early. We can use crowed sourcing or cloud software testing service to test software in a fast way.
5) Delegate the team – Test team lead can engage the testers, remove obstacles, teach them micro managing and encourage the progress.
6) Build integrity – End customer needs well-tested error free software; it also means all different components of the software are well tested before giving the software to the customer.
7) See the whole – Software is not a small part of any application; it plays the key role in big projects as well. Think globally in terms of the software relation with other parts.

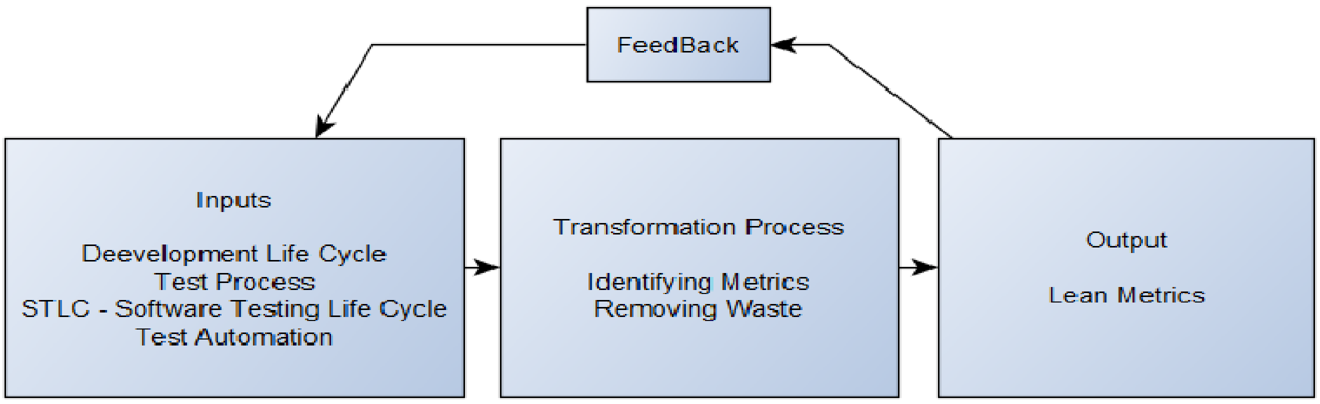

Fig. 7. Prototype for finding the right test metrics for lean canvas board using transformation models.

B. *Finding the Right Test Metrics for Lean Canvas Board*

The key metrics is an indicator for measuring the performance of anything. In the software testing process we can use the metrics to improve the software quality. Correct identified test metrics answers many questions. They can give us feedback for a software testing process. Software test metrics bring the values, which can be used to evaluate the testing process in different phases.

Step 1: Identification of the goals of the project. In testing we need to set a goal for the quality of the product, for example, a test coverage using manual and automation tools.

Step 2: Recognising the impact on the next level, for example, if we use automation testing in the project, we must know how much test coverage we get with the product under test.

Step 3: We have got already main test metrics, if we divide the full project into small sub-components and test them, we will get new test metrics.

Step 4: Once we identified sub-components, we need to create the baseline of what kind of testing we can do on the software.

Step 5: Once we have what kind of testing we are planning to do on the software, we can get smaller task information such as test cases, test case execution speed and many more.

Step 6: Establishing the review process for the collected test metrics. Review process gives a feedback for the correct test metrics.

C. *Utilisation of Transformation Models*

The transformation process is an activity that takes more or one process as input and adds a unique value to them in transformation process and at the end provides a more simplified output. As shown in Fig. 6, we use many types of different possibilities to develop and process the software itself. On the second level transformation takes place identifying and removing waste.

On the last stage as an output we get the lean metrics, which bring unique more accurate and usable metrics which are used further on with lean canvas.

VII. Conclusion and Future Research

The study explains the new approach and new possibilities for investigating and utilising lean and business canvas transformation models for identifying the right metrics in different software development lifecycles, software test process and in software testing life cycles to improve the software quality.

In order to continue research, it is necessary to accomplish the following activities:

- To carry out an experiment identifying the lean metrics in the testing process;
- To define the criteria that evaluate lean identified metrics;
- To carry out more investigation on more appropriate transformation models, if necessary;
- To utilise canvas board based on identified and collected metrics;
- To investigate and design new algorithms to improve the lean metrics identification process;
- To develop tools and framework for developing canvas board that fits for the testing process.

The new approach proposed in this article focuses on lean metrics identification and utilisation. The authors' ambition is that a new approach will generate new ideas, and some new ideas will be extracted from the article. It is an endless search in the software development and testing area to find the right test metrics which can further be adopted for testing lean canvas.

References

[1] M. Ide, Y. Amagai, M. Aoyama, Y. Kikushima, "A Lean Design Methodology for Business Models and Its Application to IoT Business Model Development," in *Agile Conference*, AGILE, 2015, pp. 107–111. http://dx.doi.org/10.1109/Agile.2015.8

[2] R. Korosec, R. Pfarrhofer, "Supporting the Transition to an Agile Test Matrix," in *2015 IEEE 8th Int. Conf. on Software Testing, Verification and Validation (ICST),* 2015, pp. 13–16. http://dx.doi.org/10.1109/ICST.2015.7102632

[3] U. Viswanath, "Lean Transformation How Lean Helped to Achieve Quality, Cost and Schedule: Case Study in a Multi Location Product Development Team," in *IEEE 9th Int. Conf. on Global Software Engineering*, ICGSE, 2014, pp. 95–99. http://dx.doi.org/10.1109/ICGSE.2014.13

[4] A. C. Barus, D. I. P. Hutasoit, J. H. Siringoringo, Y. A. Siahaan, “White box testing tool prototype development,” in *2015 Int. Conf. Electrical Engineering and Informatics*, ICEEI, 2015, pp. 417–422. http://dx.doi.org/10.1109/ICEEI.2015.7352537

[5] K. M. Mustafa, R. E. Al-Qutaish, M. I. Muhairat, “Classification of Software Testing Tools Based on the Software Testing Methods,” in *Second Int. Conf. on Computer and Electrical Engineering*, ICCEE '09, 2009, pp. 229–233. http://dx.doi.org/10.1109/ICCEE.2009.9

[6] Z. W. Hui, R. Chen, S. Huang, B. Hu, “GUI regression testing based on function-diagram,” in *2010 IEEE Int. Conf. on Intelligent Computing and Intelligent Systems*, ICIS, 2010, pp. 559–563. http://dx.doi.org/10.1109/ICICISYS.2010.5658394

[7] C. Rankin, “The Software Testing Automation Framework,” *IBM Systems J.*, vol. 41, 2002, pp. 126. http://dx.doi.org/10.1147/sj.411.0126

[8] N. P. Singh, R. Mishra, M. K. Debbarma, S. Sachan, “The review: Lifecycle of object-oriented software testing,” in *2011 3rd Int. Conf. Electronics Computer Technology*, ICECT, 2011, pp. 52–56. http://dx.doi.org/10.1109/icectech.2011.5941799

[9] D. Jeffrey, N. Gupta, “Test Case Prioritization Using Relevant Slices,” in *30th Annu. Int. Computer Software and Applications Conf.*, COMPSAC'06, 2006, vol. 1, pp. 411–420. http://dx.doi.org/10.1109/COMPSAC.2006.80

[10] G. Abu, L. Chacko, J. W. Cangussu, “Software test process control: status and future directions,” in *Proc. of the 28th Annu. Int. Computer Software and Applications Conf.*, COMPSAC, 2004, pp. 160–163. http://dx.doi.org/10.1109/CMPSAC.2004.1342701

[11] I. Burnstein, T. Suwanassart, R. Carlson, “Developing a Testing Maturity Model for software test process evaluation and improvement,” in *Proc. Int. Test Conf.*, 1996, pp. 581–589. http://dx.doi.org/10.1109/TEST.1996.557106

[12] J. Kasurinen, “Elaborating Software Test Processes and Strategies,” in *2010 Third Int. Conf. on Software Testing, Verification and Validation*, 2010, pp. 355–358. http://dx.doi.org/10.1109/ICST.2010.25

[13] P. Klindee, N. Prompoon, “Test cases prioritization for software regression testing using analytic hierarchy process,” in *2015 12th Int. Joint Conf. on Computer Science and Software Engineering*, JCSSE, 2015, pp. 168–173. http://dx.doi.org/10.1109/jcsse.2015.7219790

[14] L. Xue-Mei, G. Guochang, L. Yong-Po W. Ji, “Research and Implementation of Knowledge Management Methods in Software Testing Process,” in *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 7, 2009, pp. 739–743. http://dx.doi.org/10.1109/CSIE.2009.360

[15] S. Raman, “Lean software development: is it feasible?,” in *The AIAA/IEEE/SAE Digital Avionics Systems Conference*, 1998, pp. C13/1–C13/8. http://dx.doi.org/10.1109/dasc.1998.741480

[16] K. Ghane, “A model and system for applying Lean Six sigma to agile software development using hybrid simulation,” in *2014 IEEE Int. Technology Management Conf.*, ITMC, 2014, pp. 1–4. http://dx.doi.org/10.1109/itmc.2014.6918594

[17] J. Trimble, C. Webster, “From Traditional, to Lean, to Agile Development: Finding the Optimal Software Engineering Cycle,” in *2013 46th Hawaii Int. Conf. System Sciences*, HICSS, 2013, pp. 4826–4833. http://dx.doi.org/10.1109/hicss.2013.237

[18] C. A. Cois, J. Yankel, A. Connell, “Modern DevOps: Optimizing software development through effective system interactions,” in *2014 IEEE Int. Professional Communication Conf.*, IPCC, 2014, pp. 1–7. http://dx.doi.org/10.1109/ipcc.2014.7020388

Padmaraj Nidagundi is a founder of company IITMINDS in India and professional certified software test engineer. Currently he is also a Doctoral Student at the Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems, Riga Technical University. He obtained the Diploma in Computer Science (2004), Bachelor of Engineering, Information Science (2010), Master of Engineering Sciences in Computer Systems (2014). His research areas are software development and software testing process optimisation using lean canvas methods.
E-mail: padmaraj.nidagundi@gmail.com

Leonids Novickis is the Head of the Division of Software Engineering. He obtained Dr. sc. ing. degree in 1980 and Dr. habil. sc. ing. degree in 1990 from the Latvian Academy of Sciences. Since 1994, he has been regularly involved in different EU-funded projects: AMCAI (INCO COPERNICUS, 1995–1997) – WP leader; DAMAC-HP (INCO2, 1998–2000), BALTPORTS-IT (FP5, 2001–2003), eLOGMAR-M (FP6, 2004–2006) – scientific coordinator; IST4Balt (FP6, 2004–2007), UNITE (FP6, 2006–2008) and BONITA (INTERREG, 2008–2012) – RTU coordinator; LOGIS, LOGIS-Mobile and SocSimNet (Leonardo da Vinci) – partner. He was an independent expert of IST and Research for SMEs in FP6, FP7. He is a corresponding member of the Latvian Academy of Sciences. His research fields include applied software system development, business process modelling, eLogistics, international cooperation, web-based applications. He is the coordinator of FP7 eINTERASIA project.
E-mail: lnovickis@gmail.com