DE GRUYTER OPEN
RIGA TECHNICAL UNIVERSITY
Applied Computer Systems
ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
May 2017, vol. 21, pp. 5–12
doi: 10.1515/acss-2017-0001
https://www.degruyter.com/view/j/acss

Model-Driven Approach and Library of Reusable Source Code for Automation of IT Operations

Artūrs Bartusevičs[1], Andrejs Lesovskis[2], Viktorija Ponomarenko[3]
[1–3] *Riga Technical University, Latvia*

***Abstract* – Large software development projects with high levels of agility require several IT operations: software configuration management, bug tracking management, making software builds and deployments. Due to high agility in projects, the starting phases are very chaotic and sometimes in a few days customer is willing to get the first release of software. It means that all IT operations should be automated as soon as possible. The study presents a model-driven approach for automation of IT operations through the reuse of the existing source code. In addition, it presents a method for the development of library of reusable source code. The paper contains a brief description of the model-driven approach, library of source code and meta-models developed for a new methodology. The paper ends with the results of the practical experiments and conclusions on how this approach could be improved in the future.**

***Keywords* – Automation, IT operations, meta-models, model-driven approach, reusable source code.**

I. Introduction

Let us imagine a modern software development company that develops large and complex software where different technologies are integrated together: Oracle, Java, Ruby, .NET, etc. Multiple bug tracking systems are used to manage the development of the mentioned software and a sophisticated procedure of software configuration management should be applied to manage the source codes in multiple repositories. These repositories are controlled by different version control systems such as Git and Subversion. Frequent builds and deployments are required to support up-to-date testing process.

First, it is extremely important to automate all these activities that in the context of the current paper will also be called IT operations. Secondly, this automation should be implemented as soon as possible because manual work will waste time and human factors can greatly increase the likelihood of error. Nowadays starts of new projects are like explosion and customer is willing to get the first release of software just in a few days. However, an automated process which can prepare release is still not ready.

The mentioned software development company has a number of modern tools to automate IT operations, including but not limited to software configuration management, bug tracking, creation of software builds and deployments, continuous integration, etc. [1]. Is it possible to manage all these operations by a click of one button? For example, the project described above has a tool and a script to manage versions of the source code, a tool to make builds and deployments, and scripts to manage issues in the bug tracking system. On the one hand, it seems that all operations are automated, but in reality they are automated separately and could not be managed by one click. Therefore, a software configuration manager should first manage branches using version control tools and scripts, then make software builds and deployments for test environments, and finally update the information about related issues in the bug tracking systems.

Sometimes it is possible to achieve such a high level of automation that allows for the management of all the mentioned operations with just one click. However, are these automation solutions reusable? How much time will it take to implement similar automation in a new project? Are automations of different operations integrated together and could they be controlled by one click? Usually, these questions become a challenge for software development companies. The current paper will describe some of possible answers to the mentioned questions. Figure 1 summarises the scope of the paper and the problems that will be justified.

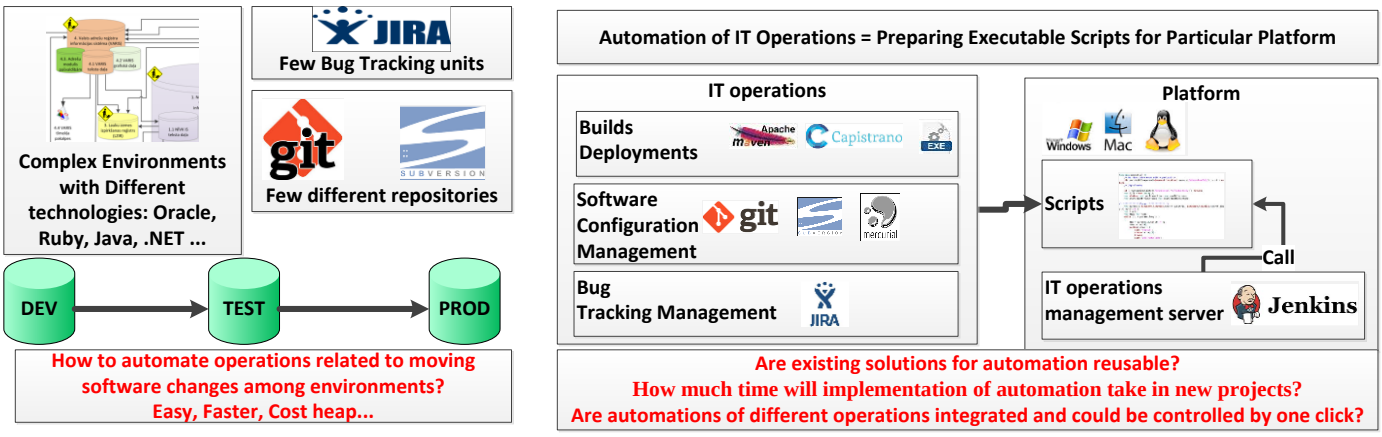

Fig. 1. The scope, definitions and problems of the paper.

Scope

The study is devoted to large-scale software development projects with complex environments, many development technologies and different tools to manage the configurations of software items and to track issues and changes in the software. The main topic is easy-to-use, fast, and cost-efficient automation of IT operations in such a large project.

Definitions

- IT operations – all technical operations to manage configurations of software items, branch using version control tools, bug track, perform continuous integration, make and deploy software builds, etc.
- Automation of IT operations – preparation of scripts for a particular platform to automate the activities mentioned in the definition above.

Problem statements

- Implementation of automation of IT operations in new projects takes too much time;
- Existing scripts, tools, and approaches for automation are not reusable in a particular enterprise; as a result, implementation time could not be decreased by reuse of existing scripts.
- Automation could not be managed by one click from one tool, switching to different tools increases overall time spent to support IT operations.

Novelty of the research

- A method for the development of library of reusable source code for automation of different IT operations;
- Model-driven approach to generate source code for automation of operations in the projects using the library of reusable source code;
- Meta-models for models, which implement a new model-driven approach provided in the paper.

Structure of the paper

The second section of the paper provides background and history of current research, as well as introduces other related works. The third section briefly describes the method for the development of library of reusable source code for automation. The fourth section contains the description of a model-driven approach for generating source code for automation and related meta-model. The fifth section is dedicated to the practical applications of the provided approach and lessons learnt. The paper ends with conclusions and areas for future research.

II. Background and History

Current research started four years ago. Then only software configuration management was included in the research topic. First, the analysis of the books about the best practices of software configuration management [2], [3] helped the authors to discover an interesting problem. Sometimes a ready solution for automation of software configuration management process is not in compliance with the initial requirements of the mentioned process. It means that the purpose of some requirements is lost during the development of process automation source code. Authors of the mentioned books [2], [3] introduce the use of models for the initial requirements of process, for example, branching models, models of software builds and deployment process, etc. After the source code for the automation of the mentioned processes is ready, it should be checked for compliance with models of requirements. It was one of the first attempts to use a model-driven approach in the field of software configuration management. Later other researchers came up with new ideas to use model-driven approaches for software configuration management and automation of related processes [4], [5]. Some of the benefits of using a model-driven approach for the automation are the following:

- Generating source code for automation by a model-driven approach could reduce manual efforts and save time during the development of code;
- Increasing traceability between initial requirements and source code.

Later, ideas to use a model-driven approach have been provided in a few papers related to software configuration management [6], [7], [8]. Until 2009, software configuration

management and automation of related processes have been studied together with build and deployment management, integration with bug tracking, continuous integration, etc. All these processes have been designated as IT operations, but huge methodology and research topic related to automation and improvement of the mentioned operations have called DevOps [1], [9]. One of the main challenges of DevOps approach was a statement that a speed of IT operations should be quite high to get in time working software to a customer during a high level of agility [9], [10]. Most of modern IT operation automation tools use a model-driven approach [1]. As an example, OpenMake tools could be mentioned. Tracy Ragan, one of the authors of OpenMake tools, said that novel solutions for automation of IT operations would not exist without a model-driven approach because static scripts could not be successfully applied in a cloud where there was no information about static addresses of servers or about platforms [9]. During the research described in the present paper, a number of tools and approaches related to automation have been studied. These tools have the following disadvantages:

- Sometimes, the scope of tools is only one particular IT operation, for example, building and deployment. There are no recommendations on how to integrate these tools with the tools that automate some other operations.

Some tools require purchasing of licences, installing a number of additional tools and refactoring a structure of the existing projects. It could be a problem for managers of companies, because usually companies already have a number of trusted tools, scripts, and best practices. Companies would like to increase reuse of the existing and trusted tools and scripts, instead of buying new, unknown, and untrusted tools.

As a result of the analysis, the authors of this paper have also taken a model-driven approach as the main idea for a new solution for automation. However, the provided solution does not have any restrictions on what tools will be used or what operations will be automated. The provided solution allows developers to create a library with the reusable units of source code for automation of different operations. The library can be extended by adding new units and can work with the existing and trusted tools and scripts. In addition, a new model-driven approach can generate source codes for automation of the operations in the particular project using the library of reusable source code.

The authors have made the first attempt to adopt a model-driven approach for software configuration management in 2014 [11]. Concept described in [12] contains names of models and steps to generate source code for automation. However, meta-models have not been developed yet. The first meta-model for this concept has been described in [12]. The first completed solution for automation of software configuration management has been introduced in [13], but the first practical results have been described in [14] and [15]. In the present research, automation of software configuration management is extended by other IT operations; in addition, the library of reusable source code and meta-models are improved and several errors from previous versions are fixed. Figure 2 illustrates the main principles of the provided approach. There is a method how to develop the library of reusable source code and a model-driven approach which could generate source code for automation of IT operations in particular projects.

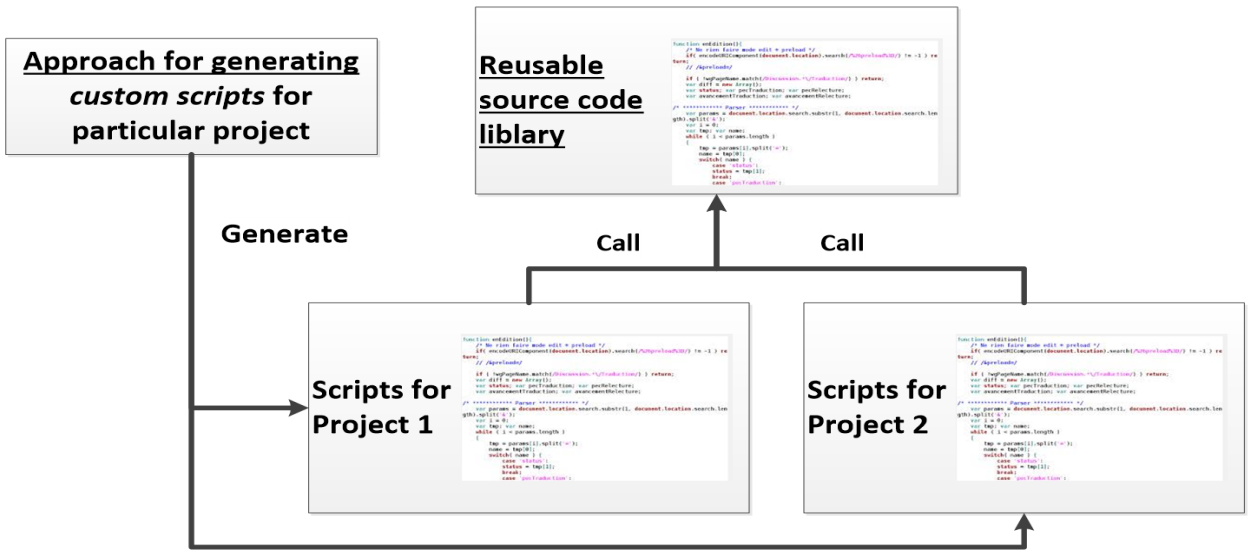

Fig. 2. The main principles of the provided approach.

III. Library of Reusable Source Code

Main Principles

The basic element of library of reusable source code is Action. Action is an executable function which gets input parameters and returns result of execution as a set of output parameters. Single Action is illustrated in Fig. 3.

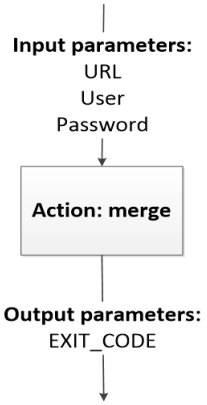

Fig. 3. Action.

Action does not contain any information about other Actions. However, Actions could be combined into ActionFlow – a set of single Actions with a particular goal. For example, ActionFlow could move software changes between two environments: DEV and TEST. Such ActionFlow could contain the following Actions:

- Finding new commits in a version control repository (find_commits);
- Merging new commits to a branch of TEST environment (merge);
- Making a software build from a TEST branch (build);
- Deploying the created build to the servers of TEST environments (deploy);
- Sending notifications to the testers about a new version in a TEST environment (notify).

Each mentioned Action receives a set of parameters, makes necessary activities, and returns a set of output parameters. The next Action depends on the output parameters of previous Action. Figure 4 demonstrates an example with the Actions build and install, which are combined to one ActionFlow.

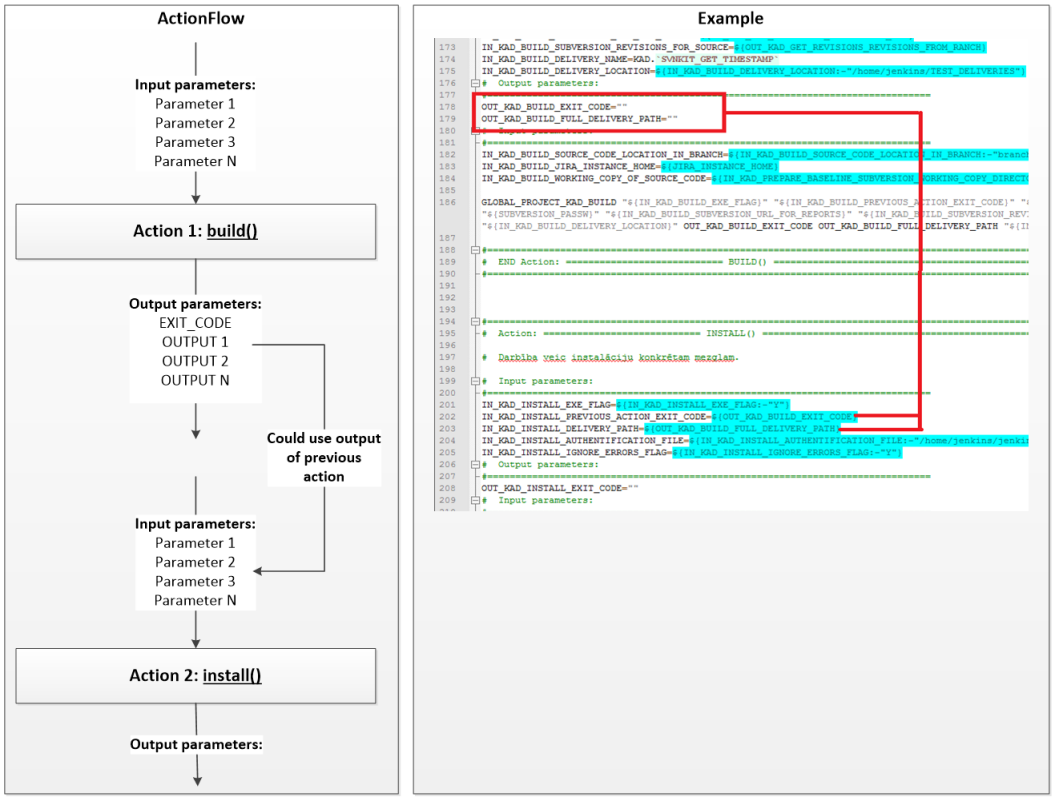

Fig. 4. Actions and ActionFlow.

The library of reusable source code contains two parts: the structure of directories and files with reusable source code and library manager. Structure of directories or library of source code contains all Actions (executable functions). Functions are structured by:

- Platform;
- Tools and Frameworks.

All Actions or functions related to a particular tool or framework are grouped in one file depending on rules of a particular programming language. The file contains private functions for internal use only and public functions for external use, called Actions in the context of the described approach. Library manager contains the following parts:

- ALL_Actions – table with all Actions. The table contains ID of each Action, description of parameters and body of Action;
- Recommended_ActionFlow – recommendations about Actions, which could be better to include in flows depending on tools. In other words, there is a collection of best practices in a particular enterprise, where the approach will be applied.
- Examples_from_existing_projects – links to automation scripts from other projects;
- Full structure of library illustrated in Fig. 5.

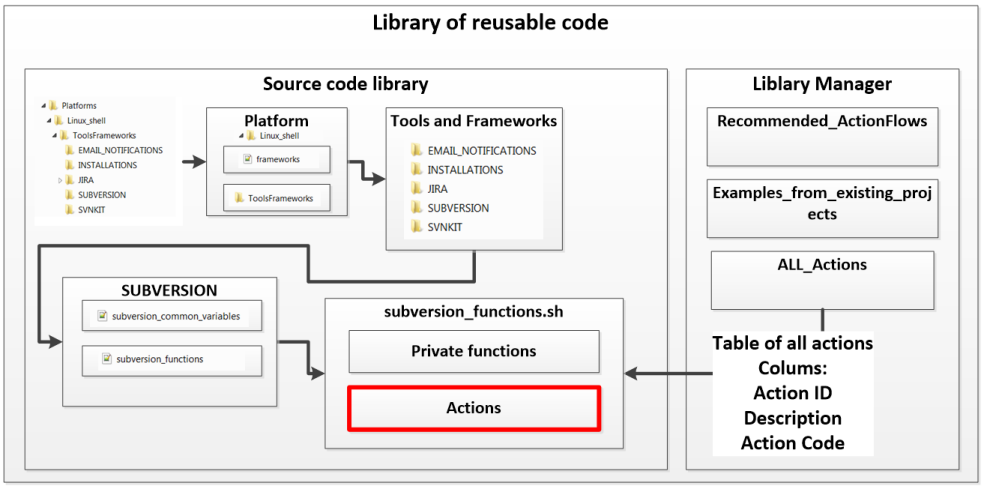

Fig. 5. Structure of library of reusable source code.

IV. Model-Driven Approach For Generating Source Code for Automation

A. Steps of Model-driven Approach

The model-driven approach provided in the present paper uses the library of reusable source code to generate scripts for automation of IT operations in particular projects. Overview of the provided approach is illustrated in Fig. 6.

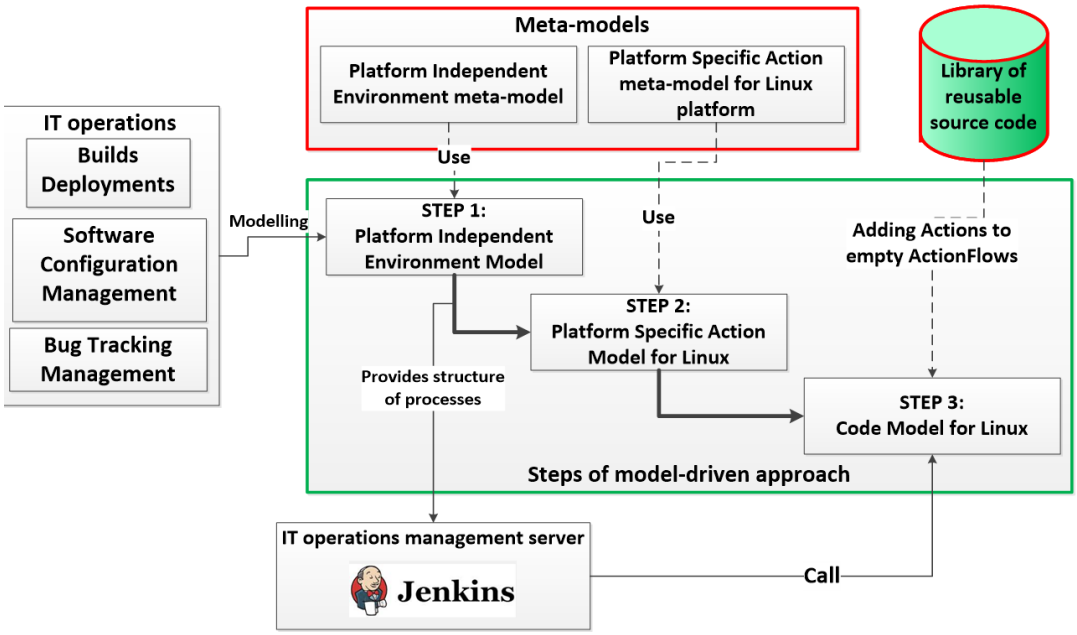

Fig. 6. Model-driven approach to generate source code for automation.

The following elements have been developed for the implementation of a model-driven approach depicted in Fig. 6:

- Library of reusable source code described in the previous section;
- Meta-model for Platform Independent Environment Model;
- Meta-model for Platform Specific Action Model for Linux platform.

There are two main goals of the approach:

- Getting source code for automation of particular IT operations;
- Getting structure of views and jobs for an automation management server (for example, Jenkins, which can be seen in Fig. 6).

To achieve these goals, the model-driven approach has been designed, which contains the following three steps:

STEP 1: Software configuration manager or other responsible person models IT operations in a particular project. The model describes environments and operations related to moving software changes between them. The model does not contain any details about the platform.

STEP 2: Platform Independent Environment Model transforming to a Platform Specific Action Model according to the rules of a particular platform and programming language. The current paper presents an action model for Linux platform; however, the provided model-driven approach could be fulfilled by action models for other platforms. Platform Specific Action Model contains empty ActionFlows described before.

STEP 3: Software configuration manager or other responsible person working with reusable source code library to add code of Actions to ActionFlows on a Platform Specific Action Model. As a result, source code for automation of modelled IT operations is ready.

Finally, the structure of views and jobs on a management server (for example, Jenkins) should be prepared according to a Platform Independent Action Model. The mentioned server will call the source code prepared by a model-driven approach to manage automation of IT operations.

B. Use Case

To illustrate the practical application of models described in a model-driven approach, let us take software development project in the development phase. There are two applications: Oracle EBS and Ruby on Rails web-portal that are integrated together. There are two bug tracking projects in JIRA, which manages changes of Oracle EBS and Ruby applications. Activities in development take place in DEV environment, but all changes from development should be transferred to a TEST environment for the testing process. In addition, statuses of related JIRA issues should be updated after each new release in the TEST environment. The main goal is to get one magic button in Jenkins server to automate all the mentioned activities related to change transfer between DEV and TEST and related to information update in the bug tracking projects. Jenkins button requires Linux shell scripts to automate the mentioned operations. This section describes how to get these scripts.

Automation of the described use case is illustrated in Fig. 7. Software configuration manager decided that automation of daily upgrades of TEST environment should be managed by "test_delivery" view and process "DEV_TO_TEST". Process should contain the following three steps:

- GET_ISSUES – find all issues in two JIRA projects, which should be updated after a new release in a TEST environment;
- COLLECT_AND_INSTALL – make builds and deployments for Oracle EBS and Ruby applications in TEST environment;
- CHANGE_STATUSES – update related issues in JIRA project after successful deployments in TEST environment.

First, a configuration manager creates a Platform Independent Environment Model, which contains information about TEST environment (applications and JIRA instances), process "DEV_TO_TEST" and steps of it.

Secondly, the model is transformed to a Platform Specific Action Model according to the designed meta-model for Linux platform. Action model actually is a source code for Linux platform, but during the current step, ActionFlows for nodes and instances are empty.

Finally, a configuration manager works with the library of reusable source code to add Actions to the corresponding ActionFlows. After that the source code is ready for Jenkins server, which is setup according to a Platform Independent Action Model (Fig. 7).

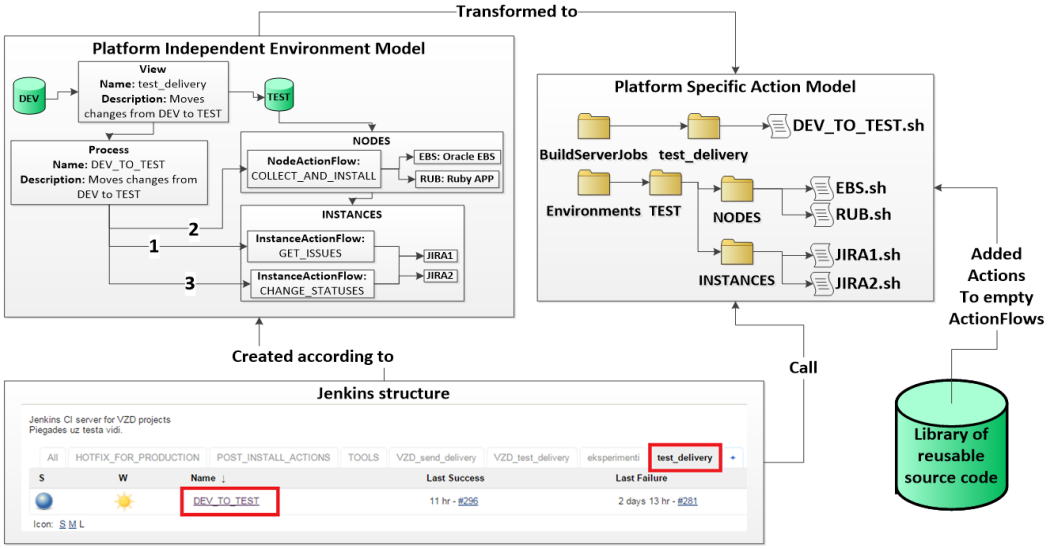

Fig. 7. Use case for a model-driven approach.

V. Experiments

During the testing of the proposed model-driven approach, automation of IT operations has been implemented in five different software development projects. Implementation time has been fixed for each project. Then the mentioned implementation time has been compared with related implementation time of automation but without the use of a model-driven approach. Figure 8 shows the difference in automation time when using old methods and the proposed model driven approach.

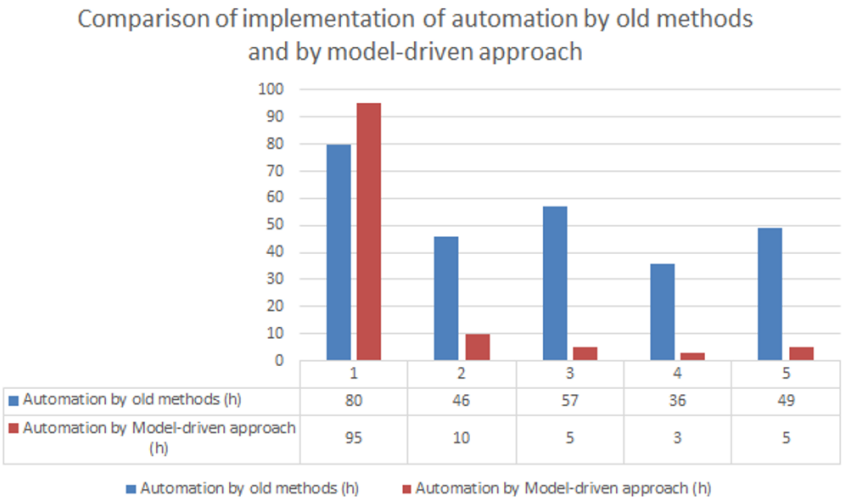

Fig. 8. Use case for a model-driven approach.

Blue colour in Figure 8 represents implementation time using old methods, and red colour represents implementation time when a model-driven approach is used. Results of the experiments show that if the library of reusable source code is empty, implementation of automation using the proposed model-driven approach is not rational – it takes notably more time. However, once the library of reusable source code contains all the necessary Actions, implementation of automation using the model-driven approach can save time (project 2 – 36 hours, project 3 – 52 hours, project 4 – 3 hours, project 5 – 44 hours). To improve the efficiency of the proposed model-driven approach, it is necessary to find out how to fill the library of reusable source code as soon as possible. It will help reduce the implementation time in project 1.

VI. Conclusion And Further Research

The study presents a model-driven approach for implementation of automation of IT operations. The approach uses reusable source code for automation of single activities. A method for the development of the library of reusable source code has also been designed during the current

research. In the context of the presented model-driven approach, the following meta-models have been designed:

- Meta-model for a Platform Independent Environment Model;
- Meta-model for a Platform Specific Action Model for Linux platform;

Practical experiments with the proposed model-driven approach show that only filled library of reusable source code could bring benefits and save time comparing to implementation of automation using old methods. It means that one of the most important further studies is the generation of source code for single actions and transfer of this code to the library. It should decrease the time for library development.

Software configuration manager, using knowledge about automation domain, could manually write the source code for single Actions and add it to the library. Once the library is filled, it could be used by a model-driven approach to generate the source code for automation of operations in a particular project.

In the future, the source code for single actions should be generated by an expert system. The expert system will use the collected knowledge about a particular domain (automation domain) and human experience. Such an intellectual solution could provide a modern approach for generation of reusable source code repositories for different domains. At the same time, models for generation of custom source code could become simpler because the number of reusable functions will be quite smaller than that of elements in the traditional programming languages.

VII. Acknowledgement

The present research has partly been funded by Latvian National Research Programme "Cyber-physical Systems, Ontologies And Biophotonics for Safe & Smart City and Society" (SOPHIS) Project No.2 "Ontology-based Knowledge Engineering Technologies Suitable for Web Environment".

References

[1] R. Azoff, DevOps: Advances in Release Management and Automation, 2011. [Online]. Available: http://electric-cloud.com/wp-content/uploads/2014/06/EC-IAR_Ovum-DevOps.pdf.

[2] S. P. Berczuk and B Appleton, Software Configuration Management Patterns: Effective Teamwork. Practical Integration, Addison-Wesley Professional, 2003.

[3] R. Aiello, Configuration Management Best Practices: Practical Methods that Work in the Real World. 1st ed. Addison-Wesley Professional, 2010.

[4] C. Bird and T. Zimmermann, "Assessing the value of branches with what-if analysis," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering - FSE '12*, 2012. https://doi.org/10.1145/2393596.2393648

[5] H. Giese, A. Seibel, T. Vogel, "A Model-Driven Configuration Management System for Advanced IT Service Management," in *International Conference on Model Driven Engineering Languages and Systems (MoDELS 2009).* USA, October 4–9, 2009. IEEE Digital Library: 4th International Workshop on Models. pp. 300–310.

[6] T. Buchmann, A. Dotor, and B. Westfechtel, "MOD2-SCM: A model-driven product line for software configuration management systems," *Information and Software Technology*, vol. 55, no. 3, pp. 630–650, Mar. 2013. https://doi.org/10.1016/j.infsof.2012.07.010

[7] T. Buchmann and B. Westfechtel, "Mapping feature models onto domain models: ensuring consistency of configured domain models," *Software & Systems Modeling*, vol. 13, no. 4, pp. 1495–1527, Dec. 2012. https://doi.org/10.1007/s10270-012-0305-5

[8] F. Schwägerl, T. Buchmann, S. Uhrig, & B. Westfechtel, "Towards the integration of model-driven engineering, software product line engineering, and software configuration management," in Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, 2015, pp. 5–18. https://doi.org/10.5220/0005195000050018

[9] T. Ragan, "21st-Century DevOps--an End to the 20th-Century Practice of Writing Static Build and Deploy Scripts," *Linux Journal*, vol. 2013, issue 230, pp. 116–120, June 2013

[10] P. Grzegrzólka, "Configuration management in agile software development," in *BIR 2009 – 8th International Conference on Perspectives in Business Informatics Research*, 2014.

[11] A. Bartusevics and L. Novickis, "Model-based Approach for Implementation of Software Configuration Management Process," in *MODELSWARD 2015: Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development*, France, Angers, 9–11 February, 2015. Lisbon: SciTePress, 2015, pp. 177–184. ISBN 978-989-758-083-3.

[12] A. Bartusevics, L. Novickis, and E. Bluemel, "Intellectual Model-Based Configuration Management Conception," Applied Computer Systems, vol. 15, no. 1, pp. 22–27, Jan. 2014. https://doi.org/10.2478/acss-2014-0003

[13] L. Novickis and A. Bartusevics, "Model-Driven Software Configuration Management and Environment Model," in Recent Advances in Electrical and Electronic Engineering: Proceedings of the 3rd International Conference on Systems, Communications, Computers and Applications (CSCCA"14), Italy, Florence, 22–24 November, 2014. Florence: WSEAS Press, 2014, pp. 132–140. ISBN 978-960-474-399-5. ISSN 1790-5117.

[14] A. Bartusevics, A. Lesovskis, and L. Novickis, "Semantic Web Technologies and Model-Driven Approach for the Development and Configuration Management of Intelligent Web-Based Systems," in *Proceedings of the 2015 International Conference on Circuits, Systems, Signal Processing, Communications and Computers*, Austria, Vienna, 15–17 March, 2015. Vienna: 2015, pp. 32–39. ISBN 978-1-61804-285-9. ISSN 1790-5117.

[15] A. Bartusevics, L. Novickis, and S. Leye, "Models and Methods of Software Configuration Management," Applied Computer Systems, vol. 17, no. 1, Jan. 2015. pp. 53–59. https://doi.org/10.1515/acss-2015-0008

Artūrs Bartusevičs is a Senior Researcher of the Institute of Applied Computer System. He obtained *Dr. sc. ing.* degree in systems analysis, modelling and design from Riga Technical University in 2015 He takes active part in several national research programmes: Latvian State Research Programme in information technologies based on ontologies and models transformation, Latvian Research Council Project in Web Technologies and Artificial Intelligence etc. He successfully combines academic and business activities: he is also a Software Configuration Management Specialist at international company Tieto Latvia.
E-mail: arturs.bartusevics@rtu.lv

Andrejs Lesovskis is a Doctoral student at Riga Technical University, the Faculty of Computer Science and Information Technology. He obtained *MSc* degree in Computer Science and Information Technology at Riga Technical University in 2009. His research areas include e-learning and semantic web.
E-mail: andrejs.lesovskis@rtu.lv

Viktorija Ponomarenko is a Doctoral student at Riga Technical University. She holds a Master degree in business informatics from RTU. As a Researcher of the Institute of ACS, She was involved in several European and National projects: European Fund Development project "Development of Insurance Distributed Software Based on Intelligent Agents, Modelling and Web Technologies", Latvian Research Council Project in Web Technologies and Artificial Intelligence, Latvian State Research Programme in information technologies based on ontologies and models transformation, FP7 project eINTERASIA etc.
E-mail: viktorija.ponomarenko@rtu.lv