RIGA TECHNICAL UNIVERSITY
Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems

Arturs BARTUSEVICS
PhD student of doctoral study program "Computer Systems"

THE DEVELOPMENT AND IMPLEMENTATION OF MODEL–DRIVEN SOFTWARE CONFIGURATION MANAGEMENT SOLUTIONS

Summary of the Doctoral Thesis

Scientific Supervisor
Dr. habil. sc. ing., Professor
LEONĪDS NOVICKIS

RTU Press
Riga 2015

Bartusevics A. The Development and Implementation of Model–driven Software Configuration Management Solutions. Summary of the Doctoral Thesis. — R.: RTU Press, 2015. – 49 p.

Printed in accordance with the Resolution of the Council of the Institute of Applied Computer Systems, Faculty of Computer Science and Information Technology, Riga Technical University, as of April 8, 2015, Minutes No. 12300–4.1/2.

The present research has been supported in part by the Latvian National Research Pprogram SOPHIS under grant agreement No.10–4/VPP–4/11.

ISBN 978–9934–10–726–9

DOCTORAL THESIS IS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF ENGINEERING SCIENCES

To be granted the scientific degree of Doctor of Engineering Sciences, the Doctoral Thesis will be publicly defended on September 21, 2015 at the Faculty of Computer Science and Information Technology, Meza Str. 1/3, Room 202.

OFFICIAL REVIEWERS:

Professor, *Dr. habil. sc. ing.* Jānis Osis
Riga Technical University, Riga, Latvia

Professor, *Dr. sc. ing.* Artis Teilāns
Rezekne University of Applied Sciences, Latvia

Assoc. Professor, *Dr. sc. comp.* Antanas Mitašiunas
Vilnius University, Lithuania.

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Engineering Sciences is my own and does not contain any unacknowledged material from any source. I confirm that this Thesis has not been submitted to any other university for the promotion to other scientific degree.

Arturs Bartusevics …….(signature)

Date: ……

The Doctoral Thesis has been written in Latvian; it contains an introduction, 5 chapters, conclusions, bibliography, 2 appendices, 55 figures, and 30 tables. The volume of the present Thesis is 228 pages. The bibliography contains 115 reference sources.

ABBREVIATIONS

MTM	Model — Transformation — Model
EAF	Environment — Action — Framework
MDD	Model–Driven Development
MDA	Model–Driven Architecture
EM	Environment Model
PIAM	Platform Independent Action Model
PSAM	Platform Specific Action Model
SCBM	Source Code Branching Model
SM	Service Model
PIEM	Platform Independent Environment Model
CM	Code Model
CIM	Computing Independent Model
PIM	Platform Independent Model
PSM	Platform Specific Model

TABLE OF CONTENTS

INTRODUCTION

In 2009 at the "Velocity Conference" a report "10 Deploys a Day" was presented by John Allspaw and Paul Hammond. This report highlighted the problem that the sharp development of a software development methodology (Agile), cloud computing technologies, operations, which prepares software builds and tranches, is unable to timely deliver to the customer a complete product [CON 2015]. The conference is considered to be the beginning of DevOps methodology, which aims to accelerate the works of software development and installations, as well as improve the quality of it [AZO 2014]. Tracy Ragan in her article [RAG 2014] marks the modern construction and installations development tendencies of tools. Tools that support software for the build and installations must be Model Driven due to the development of cloud computing technology, static scripts can no longer provide fast and efficient software for the construction and installation in the clouds [RAG 2014]. Lately, a lot of tools have appeared on the market that support the construction and installation of the Model–Driven software, such as Serena and Open Make company's products, as well as many others [AZO 2014].

Configuration management leading specialists [УДО 2011 AIE 2010] note that today modern software development projects are developing very fast, so each new project is required to implement as soon as possible automation processes that provide quality support software construction and installations process.

At present, most of the tools focus only on the construction and installation of the software, but pay little attention to other processes that directly affect the software build. Software configuration management is a discipline that reviews all the processes that affect the software construction and proper installation of the build and delivery to the customer. As the industry's leading specialists [AIE 2010 MET 2002 УДО 2011] note, it is possible to build high–quality software from a source code only if all the configuration management processes as a whole are organized qualitatively. Thus, in this Doctoral Thesis a configuration management concept will be discussed as widely as possible to identify as many factors as possible that affect the software builds.

Analyzing the modern configuration management automation solutions and their development tendencies, it should be admitted that Solutions are oriented to Model–Driven Architecture format (MDA). First of all, Model Driven approach offered by the MDA allows reducing the human factor in the transition from requirements to implementations [OSE 2011]. Secondly, while developing cloud computing technologies, statically software build scripts no longer fit, because the solution is in the clouds and scripts cannot operate with absolute server addresses and the other related values [RAG 2014].

Topicality of the Theme

The most important result of Configuration management process is a build from the source code software that is supplied to the customer. To accomplish this configuration management discipline manages the software source code and builds from it working software.

If any of these actions take place unsuccessfully and the customer receives a non–working or low–quality software, the added value of a given software development project decreases. The quality standard of Software development industry requires the orderly and automated configuration management [AIE 2010]. In the source [УДО 2011] it is mentioned that one of the configuration management process actual certifications is the fact that CMMI (Capability Maturity Model Integration) standard configuration management process is just as important as the structured development and testing process.

The Statement of the Problem

In the 21st century, when a software development approach develops rapidly and large and complex software is designed, it is often the beginning of a new project that is similar to an explosion. Already in a few days the customer is willing to obtain the first version of the software. Meanwhile, formal and automated process that builds the software is not ready yet**.** The so–called "master factor" arises, when one particular specialist knows how to prepare the software release notes from the local workstation using only his practical skills. This situation later causes unexpected errors in the configuration management process, as well as the process becomes highly dependent on the specific human competences.

Today, there is the lack of scientifically–based approaches to configuration management process automation, which would use the formal and strictly defined path from process requirements to implementations. In addition, at the stage of implementation it should be possible to re–use existing implementations for individual parts of the process. This could speed up the configuration management process automation implementation time, because only specific parts of a given project should be developed from zero, instead of full automation implementations.

The Aim of the Doctoral Thesis

The aim of the present Thesis is to develop a Model–Driven approach and methodology for introduction of the configuration management process automation, which allows the introduction of automation to reduce the time and improve the quality of automation.

Model–Driven approach to configuration management process automation introduction shows that with the help of models it is possible to automatically obtain the source code for automation process. Models conform to the MDA (Model Driven Architecture) format. The approach defines each model configuration management objective, main tasks and operating principles. New models and methods within the methodology that implement the proposed Model Driven approach principles have been developed. They allow one to automatically obtain the source code configuration management for automation.

In the context of the present research, configuration management automation quality score is a number of erroneous software builds. Software build is the main result of the configuration management automation process; thus, the fewer number of erroneous builds corresponds to a qualitative automation process.

The Tasks of the Doctoral Thesis

To reach the aim of the Thesis, the following tasks are set:

- To explore the existing configuration management solutions for process automation implementation realizing the main problems and trends of solution development.
- To identify the main benefits and disadvantages of the most recent configuration management automation solutions that meet modern trends.
- To develop an approach, methodologies, models and techniques for automation configuration management process. The approach should focus on time reduction of automation implementation re–using the company's existing automation solutions.
- To develop a software prototype offered for model display automation.
- To develop criteria for assessing the proposed approach.
- To implement configuration management process automation in the software development projects and following detailed criteria to determine the benefits and drawbacks of the developed projects.
- Based on the results of an experiment, to determine the benefits of approach, limitations, implementation risks, as well as make recommendations on the offered approaches and models for the implementation of a software development project.

- To define the further development and improvement directions of the proposed approach.

Object and Subject of the Research

The object of the research is software development and maintenance projects.

The subject of the research is software configuration management process.

The Hypotheses of Research

Configuration management research is based on the fact that by developing sharp software development methodology, a software development project begins very rapidly compared, for example, with a waterfall methodology. This fact leads to the necessity to implement the automation of configuration management processes as quickly as possible so that the customer could receive the first version of the software as soon as possible.

Developing a new approach and models for configuration management automation, the following hypotheses are proposed:

- To reduce the time of configuration management automation implementation, one can re–use the already existing automation solutions that are already functioning in other software development projects.
- The longer the configuration management automation solutions are used in different projects, the effectively they can be re–used by introducing configuration management process automation in the new software development project.

The first hypothesis is based on the fact that it is faster to customize and configurate the existing solutions than to develop a completely new one from zero. Any development takes time, which is spent on development and testing of the solution. Re–using a solution this duration is less, because the development and retesting of a complete solution should not be repeated. If for the implementation of configuration management automation already existing solutions are re–used, only parts of the process specific to a particular project should be developed from zero.

The second hypothesis is based on the fact that in the software completely all of the errors can not be detected during a testing phase. There are errors that can be discovered only while software is exploited in real life. Solutions that automate configuration management are not an exception. Thus, the longer this solution is used in configuration management processes, the more errors and failures can be detected and make this solution more stable. As a result, the efficiency of configuration management automation solution re–use will depend on how long a given solution is used.

Research Methods

In the research, the following methods are used:

- Analysis of literature;
- Simulation and metamodeling;
- Model transformation;
- Planning and organization of experiments.

Scientific Novelty

Scientific novelty of the research is as follows:

- A new approach MTM (Model — Transformation — Model) has been developed for the implementation of configuration management process automation with the help of models, re–using already existing automation solutions.

- A new methodology EAF (Environment — Action — Framework) has been developed, which implements the new MTM approach and defines the principles and steps for the implementation configuration management automation.
- New models have been developed to display the configuration management process in the methodology framework.
- A new method has been developed to store reusable solutions of configuration management.

Theoretical Value

The theoretical value of this Thesis is as follows:

- The definition of configuration management has been analyzed and the main tasks of configuration management have been defined.
- Based on the survey of literature about configuration management tasks, the configuration management process automation has been defined.
- The existing solutions for automation configuration management have been analyzed and the solution development trends have been summarized.
- A new approach, methodology, models and method for the implementation of configuration management automation based on MDA format have been developed.
- Using MetaEdit+ tool, the modeling language has been developed, which allows implementing new concepts of MTM approach, defining models, transformations and additional elements for implementation of approach.
- It has been clarified that the Model–Driven approach for implementing a configuration management process helps toreduce the risk of human factor moving from process automation requirements to implementations.

Practical Significance

This Thesis has the following practical significance:

- The experimental software prototype has been developed, which automates the generation and transformation of EAF methodology models.
- The competence group composed for practical testing activities of EAF methodology has been established. The competence group comprised senior leaders and programmers at Tieto Latvia Ltd, whose daily work was related to configuration management processes.
- Criteria for the EDF methodology evaluation has been developed, and it has been explained how to calculate the criteria scores.
- Experiments have been conducted introducing configuration management automation in five software development and maintenance projects. Based on the results of experiment, practical benefits of the EAF methodology, differences from other configuration management automation solutions, methodology implementation risks have been defined.
- A set of practical recommendations has been developed to implement configuration management automation in the new EAF methodology.

Practical results of this Thesis can be used for software development companies that are willing to improve the efficiency of configuration management automation solution and to reduce the time needed to implement automation in new projects.

The Approbation of the Research Results

The results of the research have been reported at 10 international conferences in Latvia, Italy, Turkey, France and Austria:

- October 13, 2011. The 52nd International Scientific Conference of Riga Technical University, Riga, Latvia.
- October 12, 2012. The 53rd International Scientific Conference of Riga Technical University, Riga, Latvia.
- October 17, 2013. The 54th International Scientific Conference of Riga Technical University, Riga, Latvia.
- October 14, 2014. The 55th International Scientific Conference of Riga Technical University, Riga, Latvia.
- April 27, 2012. LLU Applied Information and Communication Technology 2012, Jelgava, Latvia.
- April 27, 2013. LLU Applied Information and Communication Technology 2013, Jelgava, Latvia.
- November 22–24, 2014. The 3rd International Conference on Systems, Communications, Computers and Applications (CSCCA"14), Florence, Italy.
- December 15–17, 2014. The 13th International Conference on Telecommunications and Informatics TELE–INFO'14, Stambul, Turkey.
- February 9–11, 2015. The 3rd International Conference on Model–Driven Engineering and Software Development MODELSWARD 2015, Angers, France.
- March 15–17, 2015. International Conference on Applied Physics, Simulation and Computers, Vienna, Austria.

The results of the research are presented in the following publications:

1. Bartusevics A., Kotovs V., Novickis L. A Method for Effective Reuse–Oriented Software Release Configuration and Its Application in Insurance Area. In: Scientific Journal of Riga Technical University. Information Tehnology and Management Science, 15th series, RTU Publishing House, 2012, Riga, Latvia, pp. 111–115. (indexed: EBSCO, VINITI, Google Scholar).
2. Bartusevics A., Kotovs V. Towards the Effective Reuse–Oriented Release Configuration Process. In: Proceedings of the 5th International Scientific Conference "Applied Information and Communication Tehnologies", 2012, Jelgava, Latvia, pp. 99–103. (indexed: EBSCO, VINITI).
3. Bartusevics A., A Methodology for Model–Driven Software Configuration Management Implementation and Support. In: Proceedings of the 6th International Scientific Conference "Applied Information and Communication Tehnologies", 2013, Jelgava, Latvia, pp. 252–258. (indexed: EBSCO, VINITI).
4. Bartusevičs, A., Novickis, L., Bluemel, E. Intellectual Model–Based Configuration Management Conception. In: Scientific Journal of Riga Technical University. Applied Computer Systems. 2014/15, pp. 22.–27. ISSN 2255–8683. e–ISSN 2255–8691. (indexed: EBSCO, VINITI, Google Scholar).
5. Bartusevičs, A., Novickis, L. Model–Driven Software Configuration Management and Environment Model. In: Recent Advances in Electrical and Electronic Engineering. In: Proceedings of the 3rd International Conference on Systems, Communications, Computers and Applications (CSCCA"14), Italy, Florence, November22–24, 2014. Italy: WSEAS Press, 2014, pp. 132–140. ISBN 978–960–474–399–5. ISSN 1790–5117. (Will indexed: SCOPUS).
6. Bartusevičs, A., Novickis, L., Lesovskis, A. Model–Driven Software Configuration Management and Semantic Web in Applied Software Development. In: Proceedings of the 13th International Conference on Telecommunications and Informatics

(TELE–INFO '14), Istanbul, Turkey December 15–17, 2014, pp. 108.–116. (Will indexed: SCOPUS).

7. Bartusevičs, A., Novickis, L. Models for Implementation of Software Configuration Management. In: Procedia Computer Science. Valmiera, Latvia: 2014, pp. 3–10. (Will indexed: SCOPUS).
8. Bartusevičs, A., Novickis, L., Leye, S. Implementation of Software Configuration Management Process by Models: Practical Experiments and Learned Lessons. In: Scientific Journal of Riga Technical University. Applied Computer Systems. No.16, 2014, RTU Press, pp. 26–32. ISSN 2255–8683. e–ISSN 2255–8691. (indexed: EBSCO, VINITI, Google Scholar).
9. Bartusevics, A., Novickis, L. Model–Based Approach for Implementation of Software Configuration Management Process. International Conference MODELSWARD 2015, France, Anxhe, 9–11 February. (Will be indexed: SCOPUS).
10. Bartusevičs, A., Novickis, L. Towards the Model–Driven Software Configuration Management Process. In: Scientific Journal of Riga Technical University. Information Technology and Management Science. Vol.17, 2014, pp. 32–38. ISSN 2255–9086. e–ISSN 2255–9094. (indexed: EBSCO, VINITI, Google Scholar).
11. Bartusevičs, A., Lesovskis, A., Novickis, L. 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, March 15–17, 2015. Vienna: 2015, pp. 32–39. ISBN 978–1–61804–285–9. ISSN 1790–5117. (Will indexed: SCOPUS)

The Structure of the Doctoral Thesis

The Doctoral Thesis consists of an introduction, five chapters, conclusions, bibliography and appendices. The volume of the Thesis is 228 pages, it contains 55 figues and 30 tables. The bibliography contains 115 reference sources.

In the introduction of this thesis, the topicality of the research is stated, the aim and tasks of the research are formulated, hypotheses are put forward, research methods are defined, scientific novelty is described and practical importance of research results is provided, as well as the approbation of research results is reflected.

In Chapter 1, the software configuration management concept is defined and configuration management main tasks are determined. Based on the literature analysis, configuration management process automation it defined. Chapter 1 analyzes the existing solutions for automation of configuration management tasks, defines the main problems and solutions in today's trends.

Chapter 2 analyzes the existing approaches and tools for automation of configuration management that uses MDA format and key principles. Based on the results of the analysis, deficiencies in existing approaches are identified. In Chapter 2, the signs of configuration management approach are given, which prevent the identified weaknesses in existing solutions.

Chapter 3 describes a newly developed MTM approach to configuration management of automation implementation with the help of models. A new EAF methodology for MTM approach implementation is offered, whose development is based on the MDA format. For the implementation of a new methodology, the author of the Doctoral Thesis defines new models and according to the meta–models configuration management process for display, as well as models of transformation laws that allow one to change patterns at the abstraction level. The methodology proposes implementing configuration management process automation, using

existing automations for certain configuration management tasks. The method for storage of existing configuration management automation solutions is developed.

In Chapter 4, a new EAF methodology is tested. Evaluation criteria of methodology are provided and the theoretical approbation are described. During the methodology testing, five software development projects have been implemented. As a result of experiment, benefits and drawbacks of methodology have been defined. By analyzing the benefits, weaknesses, reviewer feedbacks obtained by publishing the methodology theoretical foundations in inventories of scientific conferences, it has been found out that existing benefits may be increased, but the number of shortages can be reduced by making improvements in the methodology.

In Chapter 5, the development of improved version of EAF methodology is described. The main objective in development is to prevent through experiments opened weaknesses and to take into account the remarks made by the reviewers of scientific articles who got acquainted with the EDF methodology. During repeated experiments, it has been shown that the weaknesses have been prevented and the benefits increased. At the end of the chapter, deficiency rectification activities are described. Based on the comparison of results of the first and second round of experiments, methodology key benefits are defined, differences from other configuration management automation approaches are determined, as well as methodology implementation risks are identified and future development directions are defined.

In the final part of this Thesis, the main results of the research are provided, aims and tasks are substantiated, the hypotheses are proven, as well as possible future research directions are listed.

1. THE RESEARCH OF SOFTWARE CONFIGURATION MANAGEMENT

The Definition of Software Configuration Management

As a result of literary analysis [AIE 2010, BER 2003, DEP 2010, PAU 2007, MET 2002, KAN 2005, CON 2002, GLO 2012, BRU 2004, DAR 2001, WES 2005, MEL 2006, BEL 2005, VAC 2006, WIK 2013, ОРЛ 2011, ЛАП 2004, УДО 2011, ЗАМ 2008], more than 20 different definitions are found, which explain the concept of configuration management. In the found definitions, common parts have been joined and, as a result, a configuration management process has been defined.

Configuration Management software is a set of processes that identify and control software items and their process of evolution, provide guidelines for the build and installation process of software as well as make the software item status tracking.

Configuration Management software has the following main tasks:

- Item identification of configuration;
- Version control of configuration item;
- Finished product building (preparation process of tranche or installation packages (building engineering));
- The installation of finished product (deployment);
- The parallel development support (with configuration items at the same time working on a number of developers) (branching));
- Metrics collection for configuration item changes, versions and different product configurations;
- Configuration item accounting and audit.

When analyzing the literature, the main configuration management definitions and main tasks have been found out. Due to the fact that the aim of the Thesis is to develop an approach

and methodology for configuration management automation, based on the obtained information of this chapter, configuration management automation will be defined.

Solutions of Configuration management automation — software, which implements configuration management tasks defined in this chapter by minimizing human intervention. Automation is mainly focused on version control, source code management, software building development, software installation.

Thus, the wording "to develop automation for configuration management" within the framework of the research means to develop a set of software (scripts, libraries, frameworks), which with minimal human intervention is able to perform configuration management tasks defined in this chapter, mainly version control, source code management, building and installation development.

In this chapter, automation of software configuration management is defined. Automation needs to solve the following tasks: version control, source code management, product builds and installations, metrics collection.

During the research five important features have been identified, which characterize the process of modern configuration management: the process solves complex of all tasks, the process is the Model Driven, You are able to use the existing tools and scripts in the new Model–Driven automation solution, version control works not only with a code, but also with models to be able to support projects with the MDD (Model Driven Development) approach and the process does not conflict with the quality standards.

The next chapter analyzes configuration management automation solutions, which correspond to the Model–Driven Architecture format. For each approach the following factors will be evaluated:

- Compliance of approaches for Model–Driven Architecture principles;
- Access area, resolving configuration management tasks;
- The possibility to use existing tools or scripts as well as create new solutions, which can be applied repeatedly.

2. MODEL–DRIVEN SOFTWARE CONFIGURATION MANAGEMENT

General Principles of Model–Driven Architecture

Model–Driven Architecture initially has been designed for software development. Model–Driven development is application of models during software development lifecycle. MDA is related to such development methodologies, where the use of models is the main approach to obtain primary artefacts, where knowledge about software is represented by a particular modelling language.

Model in the context of MDA is the description of system or part of system using a language with strong defined syntax and semantics and this language should be readable by a computer. Each particular system could be defined by different models; however, strong relations should be defined between mentioned models (e.g., full — part, where one model defines general aspects of system, but the other model provides detailed information about a particular part of system). Nowadays MDA has strong relations with UML; this fact allows reducing risks of invalid model translations; however, domain specific languages expect some different notations instead of UML [DON 2011, OSI 2011].

MDA allows designing models with a high level of abstraction and these models are independent of particular platforms where they should be applied. The mentioned models could be stored in special centralized repositories. MDA contains the following technologies: unified modeling language (UML), metaobject facilities (MOF), interchange of XML metadata (*XML*

Metadata Interchange — XMI) and common warehouse metamodel (CWM) [DON 2011, OSI 2011].

Transformation of models is a unified process to convert a model with a particular level of abstraction to other model with a different abstraction level; however, equivalence should be saved between the mentioned models. Transformations between different models are basics of a model–driven approach. Each model could be represented as a UML diagram, OCL specification or a set of text. Model–driven approach defines a set of kinds of models. These models could be abstract (related to functionality of system) or concrete, which describe relations with a particular platform, technology or implementation. There are the following kinds of models in the MDA:

- CIM — Computation Independent Model;
- PIM — Platform Independent Model;
- PSM — Platform Specific Model;
- Code Model or ISM (Implementation Specific Model.

There are options to make the following transformations in the context of MDA approach: CIM to PIM, PIM to PSM and PSM to Code Model. It is also possible to make transformations without changing the abstraction level. Many models with a lower level of abstraction could be related to one model with a high level of abstraction, for example, from one PIM model many PSM models could be created and each PSM model will represent information for a different platform. General MDA process is provided in Fig. 2.1.

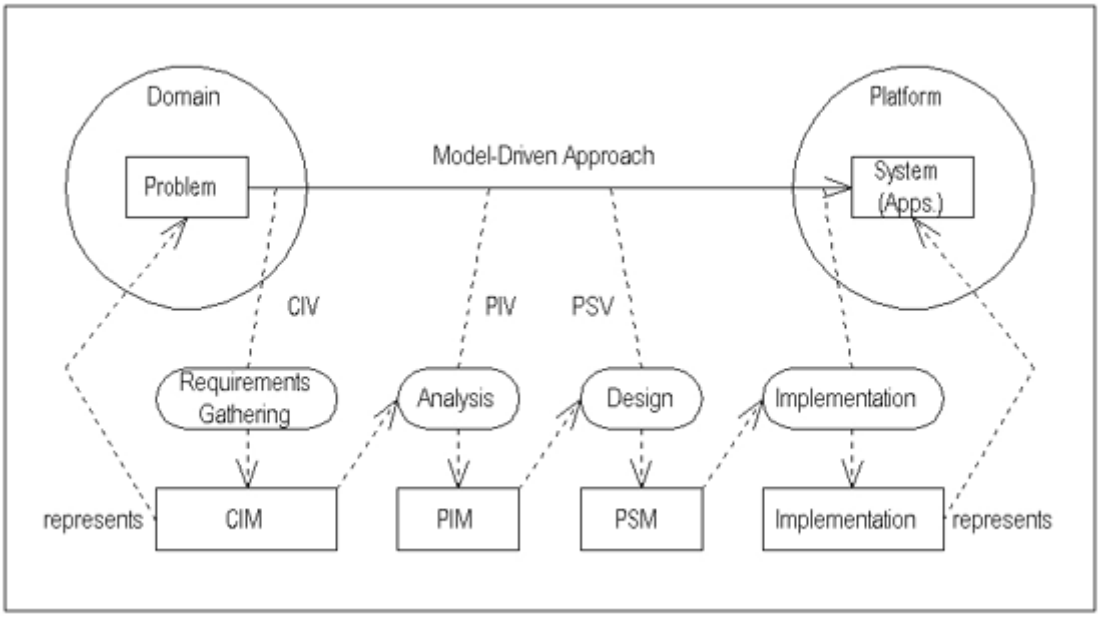


Fig. 2.1. Model–Driven Architecture.

Model–Driven Approaches for Software Configuration Management

Model–driven software configuration management [PIN 2009] contains models, a number of definitions of meta–models and recommendations on how to improve relations between configuration management and software development. The above–mentioned study describes a tool called Model–Driven Configuration Editor. This tools has been developed in the Eclipse environment using Eclipse Modeling Framework and Graphical Modeling Framework. Transformations between models are implemented by openArchitectureWare (oAW). The study provides guidelines on how to design another tool for modeling the software configuration management process.

The approach described in [PIN 2009] has the following advantages:

- Conception of merging software configuration management and model–driven development;
- The abstract model of product configuration;
- Instructions on how to improve and extend a tool for support of model–driven software configuration management.

There are the following conclusions about the approach described in the study [PIN 2009]:

- The provided approach is oriented to projects with a model–driven development approach. There are no suggestions on how to apply the provided approach in projects with classic methodologies of development, where main artefacts are a source code instead of models.
- Approach is oriented only to one of the main tasks of software configuration management — identification of software configuration items.
- Approach contains all main elements from general model–driven architecture: there are meta–models, model creation rules, implementation of PIM and PSM is provided.

The study [PIN 2009] provides an abstract approach for implementation of model–driven software configuration management. The provided approach is oriented to the identification of software items; the items could be the following:

- Hardware components;
- Software components;
- Source code files and documentation files.

Each group of the above–mentioned software configuration items contains a meta–model. From this meta–model the PIM model could be created. Finally, the PIM model could be transformed to the PSM model using special transformation rules. The main result of the mentioned modelling process is XML files, which describe configuration items and their structure. Example of PIM model is provided in Fig. 2.2.

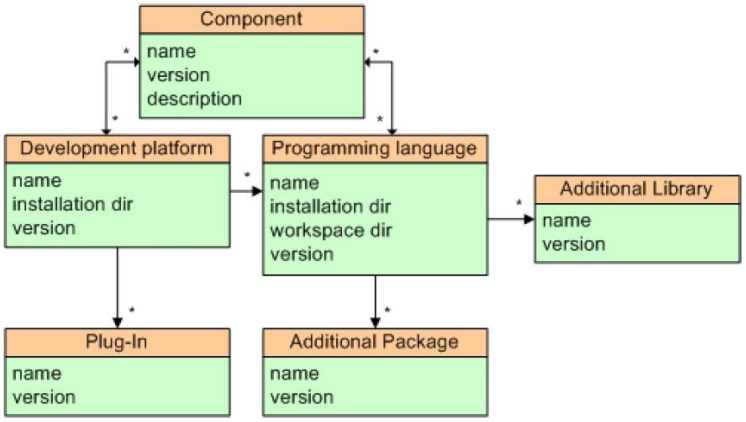


Fig. 2.2. PIM model.

Figure 2.3. represents the PSM model created by transformations from the mentioned PIM model. The PSM model is related to Eclipse IDE.

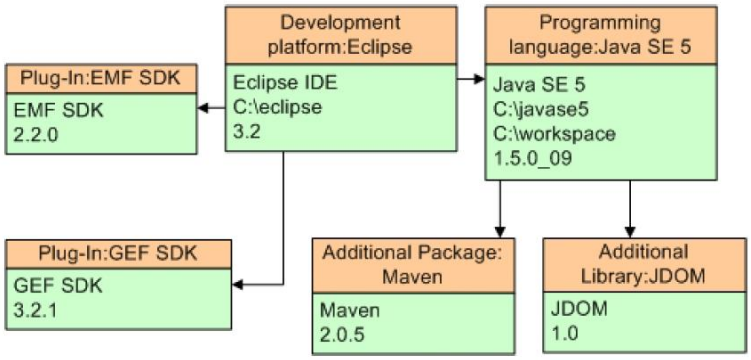


Fig. 2.3. PSM model.

Finally, the model provided in Fig. 2.3. could be transformed to an XML file, which will be parsed by a software configuration management tool.

The study [GIE 2009] describes a software configuration management process in general. The main principles of software configuration management are taken from the ITIL framework (Information Technology Infrastructure Library). Software configuration management process is represented by different components. Each component has a meta–model, which allows making an abstract software configuration management model. The next step of approach describes how to transform this abstract model to a platform specific model. The approach is based on main principles of MDA. The models allows describing a software configuration management process with a different level of abstraction to improve a general overview of process. The study describes a tool for implementation of model–driven software configuration management. The abstract model of software configuration management has been designed for ITSM (IT Service Management). Figure 2.4 provides an example of the above–mentioned model.

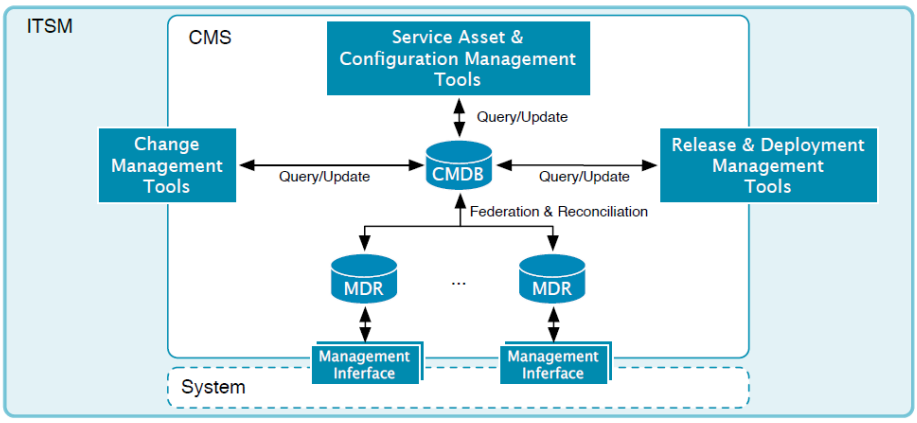


Fig. 2.4. Abstract model of software configuration management.

The approach [GIE 2009] provides the following kinds of models:

- Model of management tools;
- Model of configuration management database (CMDB);
- Model of data repository (MDR).

Relations provided in Fig. 2.4 (Query/Update) should be implemented by transformations between different kinds of models. There are no practical examples of implementation of the mentioned transformations.

The approach provided in the study [CAL 2012] is oriented to improve the integration of different tools using the software configuration management process for automation. There are a number of different tools that should be integrated together to support an end–to–end software configuration management process: version control tools, bug tracking systems, continuous integration tools, building and deployment tools etc. Usually all these tools work independently. There is a lack of integration between these tools. The authors of [CAL 2012] think that improvement of integration of the mentioned tools could improve general automation of software configuration management process. To improve the mentioned integration between different tools, the authors of [CAL 2012] provide a general concept for each kind of tools. Finally, the approach provides a task ontology for software configuration management process. This ontology could be used as a general model of software configuration management. The main scope of this model is to show integrations between different kinds of tools. Ontology is oriented to a version control task, which is one of the main tasks of software configuration management.

There are the following conclusions about the study [CAL 2012]:

- Ontology for an abstract software configuration management process is provided. The provided ontology is independent of a concrete tool or a platform.
- The provided ontology could be used as a baseline for a platform independent model. Principles of ontology are based on the elements of ISO standards and Subversion version control system.
- Ontology is oriented to the integration of different kinds of software configuration management tools.
- There are no suggestions on how to obtain the PSM model from the provided ontology.

The study [CAL 2012] is not a single attempt to apply artificial intelligent methods to automate configuration of tools using the software configuration management process. There are a number of studies related to the configuration of project management tools using artificial intelligent methods for integration of different tools [BER 2012, BER 2011].

Analyzing the Configuration Management Model–Driven solutions, approaches and their development trends [PIN 2009, GIE 2009, BUC 2009, CAL 2012, KR 2014, FIT 2014, FUG 2014, CRA 2008] have been explored, as well as the latest tools for Model–Driven approach practical realization [OPE 2014, SER 2014, AZO 2014] have been studied. The compliance of solutions for MDA format and comments are provided in Table 2.1.

Analyzing the latest tools, which introduce the process of Model–Driven configuration management [OPE 2014, SER 2014, AZO 2014], the main benefits and disadvantages found by the author of this Thesis have been summarized. The most essential achievements in tools [OPE in 2014, SER 2014 AZO 2014] are as follows:

- The majority of the analyzed tools are consistent with the key principles of Model–Driven approach. Tools allow one to relatively quickly model the configuration management process of software development project, and then implement to specific technologies and platforms.
- Configuration management process is reviewed and easily configurable thanks to the intuitive understandable users' graphical direct exposure. Configuration manager forms the build scenarios of product with mouse clicks instead of writing huge scripts.
- Tools are fully in line with modern trends in the software development industry. The possibility of establishing parallel builds has been implemented, the system has been configured, which is capable to perform several dozen builds a day. Mostly all tools have built–in functions that support the processes of build formation also in the clouds. Consequently, one no longer has to write statically scripts for each project separately.

Table 2.1

Comparison of Model–Driven Solutions for Software Configuration Management

Solution identifier	Meta–models	Models with a different level of abstraction	Transformation solutions	Tools support	Comments
[PIN 2009]	+	+	+/–	+/–	Best of all mentioned solutions in a substantive way, because there is a partial solution to the meta–model, a tool that performs model transformations.
[GIE 2009]	+/–	+/–	–	+/–	A purely theoretical solution, no specific details of how this solution can be implemented. The approach is oriented to the only one technology.
[BUC 2009]	–	A purely theoretical solution	–	+/–	The solution is oriented only to the version control rather than to the configuration management process as a whole.
[CAL 2012]	+/–	+/–	+/–	+/–	Although the solution does not comply with the general principles of Model–Driven approach, there is an important problem highlighted, which must be taken into account in the Model–Driven configuration management solution development — mutual integration of tools. At the theoretical level, as a solution it is offered to create an abstract integration model for tools that support the configuration management process.

Gathering information from sources [OPE in 2014, SER 2014 AZO 2014], it has been concluded that the tools have also disadvantages:

- Mainly all the tools are oriented to the following configuration management tasks: construction and installation management, product release note preparation to the customer and metrics collection. However, hardly any tool pays sufficient attention to management automation of source code. In turn, without the deliberative source code management construction and installation process cannot be qualitative [AIE 2010].
- By implementing the configuration management process with tools, which are mentioned in these sources [OPE 2014, SER 2014, AZO 2014], lower abstraction level models (scripts, project structure, compilation algorithms etc.) are defined. If they are ignored, the solution will not work correctly. Often, however, the company has its own specificities and approach to different script and project configuration. The company will hardly be ready to apply the solutions and approaches that have been tested for years. For example, if implementing some of the new building and installation tools, it will be necessary for all Java projects to remake classes and package structure, unlikely the company will be ready for that, while the customer most likely will not want to pay for this activity.

3. DEVELOPMENT OF MTM APPROACH AND EAF METHODOLOGY

Definition and General Description of MTM Approach

MTM (Model — Transformation — Model) — a newly developed approach to obtain the source code for automation of software configuration management process. MTM Approach provides that all configuration management processes are managed by a re–executed exit code from a special configuration management server. This exit code is obtained automatically, sequentially modeling configuration management automation processes. Models conform to the MDA format. It is intended that software development in a company, which uses MTM, has been implemented in the solution database that holds re–usable source code units for certain configuration management tasks for individual platforms. The solution database stores mentioned source code units following certain techniques that have been developed and later upgraded within this Thesis.

MTM approach ensures that in the beginning a Configuration Manager simulates the configuration management process regardless of a specific platform. Later, it is supplemented with the implementation details, which configuration manager obtains from the solution database. A platform specific model is formed for a specific configuration management process. Finally, from this model the source code is automatically generated for configuration management process automation. In Fig. 3.1, you can see the MTM approach scheme.

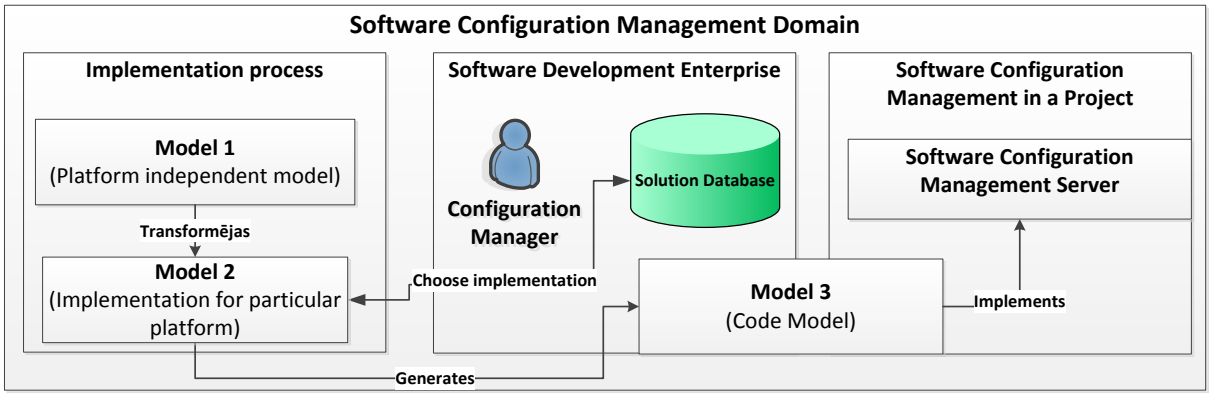


Fig. 3.1. MTM elements and relations.

EAF Methodology for Implementation of MTM Principles

Methodology objective is to define the implementation steps of configuration management automation and to provide an opportunity for a new process to use theb already existing solutions. EAF is an abbreviation of of the methodology "Environment — Action — Framework". The EAF methodology implements MTM approach principles, implementing models visible in Fig. 3.1 (Model 1, Model 2 and Model 3), as well as the solution database and model transformation rules. A gradual transition from one model to another, using the model transformation rules, defines configuration management automation source code formation steps. The re–use of solution allows reducing the implementation time of automation, thus minimizing the risk of unexpected errors that occur when all solutions are developed from scratch. Development of EAF methodology is organized by many iterations. Results of the mentioned iterations are described in papers [BAR 2012a, BAR 2012b, BAR 2013, BAR 2014f].

During the development of the EAF methodology the following concepts have been introduced:

1. **Project** — a software development project, within which configuration management is described.
2. **Company** — a specific company, which implements software development projects.
3. **Configuration Manager** — a user, who using the EDF methodology performs modeling and implements configuration management process automation in the project.
4. **Configuration management solution warehouse (SCMWarehouse)** — a structure, where all configuration management automation solutions are held within the company.
5. **The management system of configuration management solution warehouse** — an application that manages the information at SCMWarehouse.
6. **Platform** — a specific operating system, in which a configuration management process exit code is implemented.
7. **Configuration management server (SCMServer)** — a centralized server, from which the execution of configuration management exit code is managed. The server is focused on a specific platform.
8. **Environment** — a set of infrastructure, in which the developed software (application servers, databases, external system interfaces, etc.) is located. Each environment is designed for a specific activity in software development life cycle, for example, development, testing, quality acceptance testing of exploitation, etc.
9. **Action** — the activity in the configuration management automation process. Usually the activity solves one of the main configuration management tasks, such as: creation of the software build, source code management, software installation in one of media, etc.

The EAF methodology contains the following elements:

- **Environment Model metamodel** — a modeling language for the Environment Model development.
- **Environment Model (EM)** — a configuration management process model that represents all the specific project environments, among which the change occurs in the transmission of software.
- **Source Code Branching Model (SCBM)** — a model that illustrates the laws of software source code management depending on the Environment Model, shows what branches of source code correspond to what environments and in which way a source code changes transmission (merge) between the different branches.
- **Platform Independent Action Model (PIAM)** — a model that shows what actions are to be taken to transfer the software changes between instances in the Environment model. In this model, activities do not contain any implementation details and do not depend on any platforms.
- **Platform Independent Action Model metamodel** — a modeling language for the PIAM model development.
- **Platform Specific Action Model (PSAM)** — an extended variant of the PIAM model. Unlike the PIAM, this model contains all the information on the operation implementations: platform specific tools, scripts, instruction manual.
- **Service Model** — a model that shows the mutual integration tool. Model contains the pairs of tools. For each tool located in a particular pair, there is a set of functions or methods to call up the second pair of tools. Service Model is required for different tool integration description. If the PSAM model can see the tools needed for the analysis of configuration management activities, then the service model shows how tools work

with each other (integrate) to be able to maintain a full–fledged configuration management operation flow.

- **Service detection algorithm** — an algorithm that depending on the tools of PSAM model determines tool pairs, or services. During the implementation of PSAM model the configuration manager in the beginning has to implement services, which are determined by a service detection algorithm.
- **The transformation laws "E–> S" —** a set of laws that operate with the Environment model and prepare the appropriate source code branching model.
- **The transformation laws "E–> S" —** a set of laws that operate with the Environment model and prepare the appropriate PIAM model.
- **Solution Choice Module (SCMWarehouse) —** a storage, where all the company existing configuration management solutions are stored.

Figure 3.2 demonstrates the EAF methodology general framework. Activities and methodology key steps are marked by arrows. "Configuration Manager" is a user, who implements company configuration management processes and produces various models.

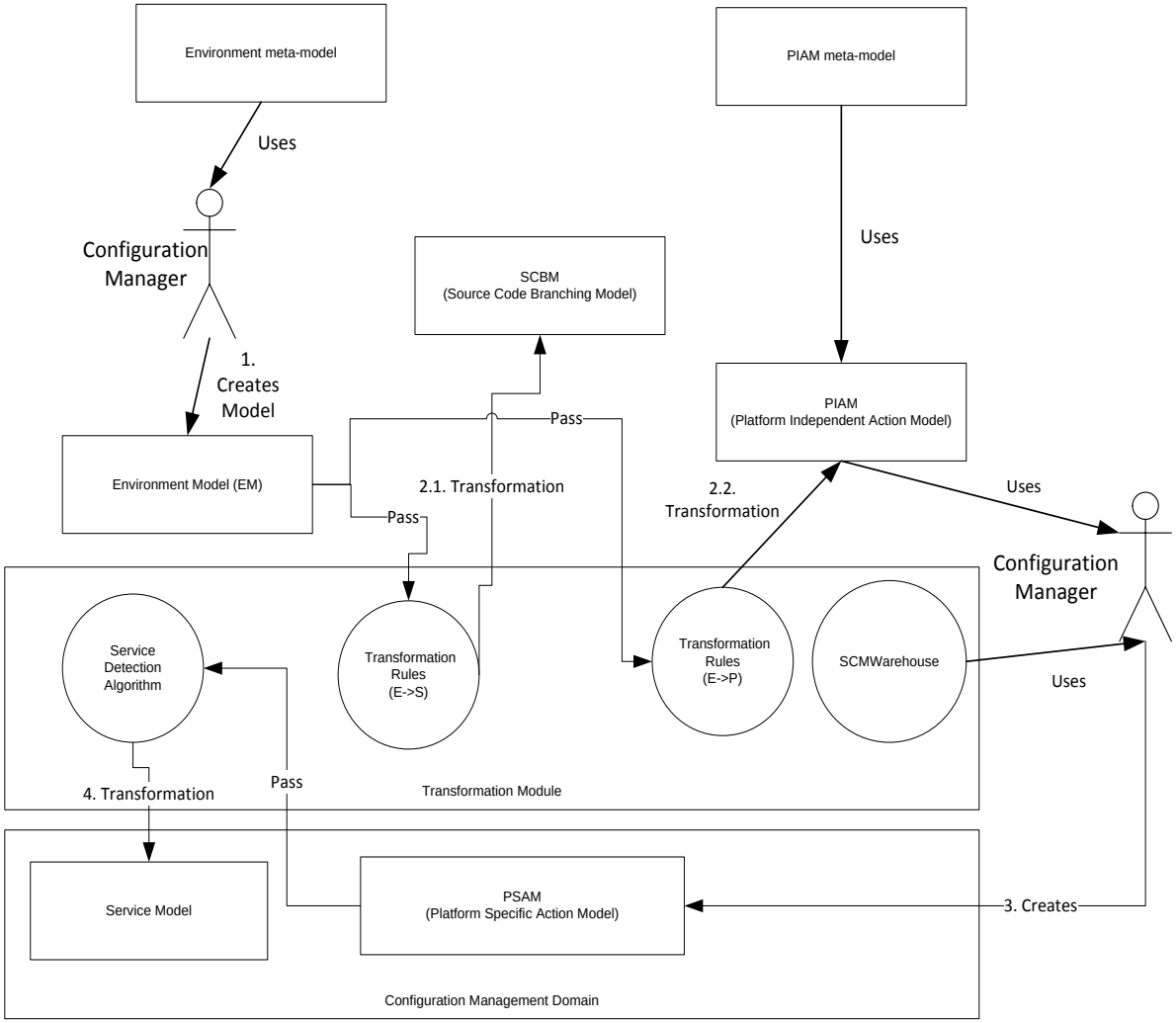


Fig. 3.2. General scheme of EAF methodology.

It is intended that the company develops the Choice Module of Solutions that in a structured manner holds all the company's solutions for configuration management activities. The EAF methodology contains four main steps:

- Configuration Manager makes the Environment Model for a specific software development project.
- Transformation laws "E–>S" and "E–>P" convert the Environment Model into SCBM and PIAM models. At this point, the user knows what source code repository branches should be built to maintain a source code base for each environment from the Environment model. Configuration management activities are also known that are required to transfer the changes between environments.
- Using activities in the PIAM model, a Configuration Manager from a Solution Choice module chooses one specific solution for each activity. As a result, the PIAM model expands to the PSAM model that contains information about platforms, tools, scripts, etc.
- PSAM model is processed by a Service detection algorithm that determines tool pairs for integration.

Finally, both the tool integration and the PSAM model are implemented in the project configuration management problem domain. The EAF methodology is ending when in a configuration management problem domain a source code management system is implemented according to the SCBM model, all integration shown in the SM model is implemented, and the PSAM model is implemented.

Meta–Model for the Environment Model

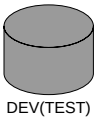
The scope of the Environment Model is to show all flows of software changes between different environment. Meta–model is a source for creation of the Environment Model. Each flow of software changes is related to a particular event. One event could have multiple flows. For example, an event called "Move changes from DEV to TEST environment" could have two flows. The first one represents a flow of changes from DEV to TEST1 environment. In this case, environment TEST1 is used only for validation of a particular software build. If this build is valid, it could be installed on TEST environment by the second flow of the mentioned event. In this case, TEST environment is used for real testing process. Only builds that are validated in TEST1 environment could be installed to TEST. The Environment Model should provide information about all flows and events related to transfer of software changes between different environments in a particular project. The Environment Model could be created by software configuration management that should make decision about environments, events and flows.

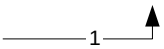
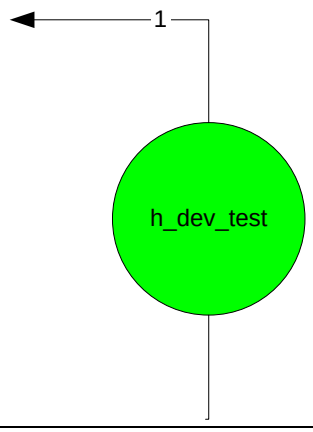
Graphical Representation of Elements and Examples

Table 3.1 contains information about elements of meta–model of the Environment Model. Each element contains a set of attributes, which should be fulfilled during the modelling process.

Table 3.1

Elements of the Environment Model

Name/Graphical representation	Attributes and description
Actor 	Software developer who makes changes in a source code. Attributes: **Name** **Description** Additional information about a developer
Environment 	Environment or instance where software is stored. Usually it is a set of infrastructure (application servers, database servers, firewalls etc.), in other words — all that needed to run particular software. Each environment has the following attributes:

	Name — name of environment. **Description** **CustumerSupportFlag** — flag that shows a particular environment supported by a customer. **DevelopmentFlag** — a flag that shows whether developers make changes in the software manually or changes could be installed only by build. In other words, this flag separates development environments from other environments (test, qa, prod, etc.). **Original environment flag.** It shows the scope of environment. If value of this attribute is "Y", it means that the environment could be used for a real process, for example, testing. If a value of this attribute is "N", it means that the environment should be used only for validation of software build or continuous integration but not for a real process. **OriginalEnvironmentName** — if a particular environment is not used for a real process, but only for validation of build, this attribute contains the name of original environment. For example, if TEST1 environment should be used to validate builds for TEST environment, the value of attribute OriginalEnvironmentName should be 'TEST'.
ConfigurationItemFlow 	Represents a way, by which changes should be transferred from one environment to other. Attributes: **Name** — a name of a particular flow and short description about the meaning of particular flow. **Sequence** — a sequence of a particular flow in the related event. **Source** — an environment, where the software changes are stored. **Goal** — an environment, where changes should be transferred. **Description** — additional information about a particular flow.
Event 	Defines an event for representing transfer of changes from one environment to some others. Attributes: **Name** **ConfigurationItemFlows** — an array of ConfigurationItemFlows. **Description** — additional information about a particular event. **AllChangesMoveFlag** — a flag that shows amount of changes, which should be transferred between environments (all new changes or only particular changes).

Meta–Model for Platform Independent Action Model

Platform independent action model (PIAM) shows all actions needed to implement all flows of changes defined by the Environment Model. The aim of PIAM model is to show all software configuration management actions and all related attributes to implement the Environment Model. Values of attributes in the PIAM model are empty, because this model is independent of any platform and technology.

Table 3.2

PIAM Actions and Attributes

Name	Identification	Description
Development of software changes	DEVELOPMENT	Simulates action of development of source code and rules related to the mentioned development.
Saving changes in version control repository	COMMIT_CHANGES	Defines rules and implementations how to save any changes in the version control system and how to manage different versions of a source code.
Preparing baseline for a particular environment	PREPARE_BASELINE	Defines the approach of management of source code, branching and merging strategies, how to prepare promotion branches, baselines etc. Provides also details about technical implementation of the mentioned approach and strategy.
Software building	COMPILE_BUILD	Defines the approach how to build software from a source code and all related implementations (scripts, tools, etc.)
Software deployment (installation)	INSTALL_BUILD	Defines the deployment approach in a particular project and its implementation.
Software delivery for a customer	PRODUCT_DELIVERY	Defines the approach how to deliver ready software to a customer, how to prepare build, installation guides and other related documentation.
Notification about updates of environment	ENV_UPDATE_NOTIFICATION	Defines environment post update actions.

Elements of PIAM meta–model are related to main principles and practices of software configuration management. Process could be decomposed to multiple tasks and implementation of these tasks should be centralized, manageable and reusable [AIE 2010, BER 2003]. PIAM model contains information not only about software configuration management tasks but also about continuous integration server or, in other words, configuration management server. This server implements centralized place where all tasks should be implemented [AIE 2010, PAU

2007, MET 2002]. Table 3.2 contains information about software configuration management actions in the context of PIAM.

Table 3.3 contains attributes of PIAM actions with a short description.

Table 3.3.

Attributes of PIAM Actions

Attribute	Description
Platform	Name of platform, where a particular action should be implemented
SolutionName	Name of solution.
NeededTools	Tools needed for implementation of a particular solution.
LocationsOfSolutions	Locations of reusable solutions related to a particular action (scripts, tools, implementation guides etc.)
Description	Additional description notes about the implementation of a particular action.

To apply all actions for software configuration management, a continuous integration server or a configuration management server should be implemented. The PIAM model allows describing such a server and contains an element called “Continuous Integration Server” with the following attributes:

- Platform — a platform name, where a configuration management server should be implemented,
- ToolName — a name of continuous integration server,
- InstallationNotes — notes about the implementation of a particular server,
- LocationOfSolutions — locations of reusable scripts to setup continuous integration server (if such scripts are).

Additional elements in the PIAM meta–model are Events. All Events should be copied from the Environment Model together with all related flows. Finally, the PIAM model shows all configuration management actions needed to apply each flow of each event from the Environment Model.

The PIAM model represents only empty attributes for each mentioned element, no details about platform and implementation are given. Figure 3.3 represents the structure of elements of the PIAM model.

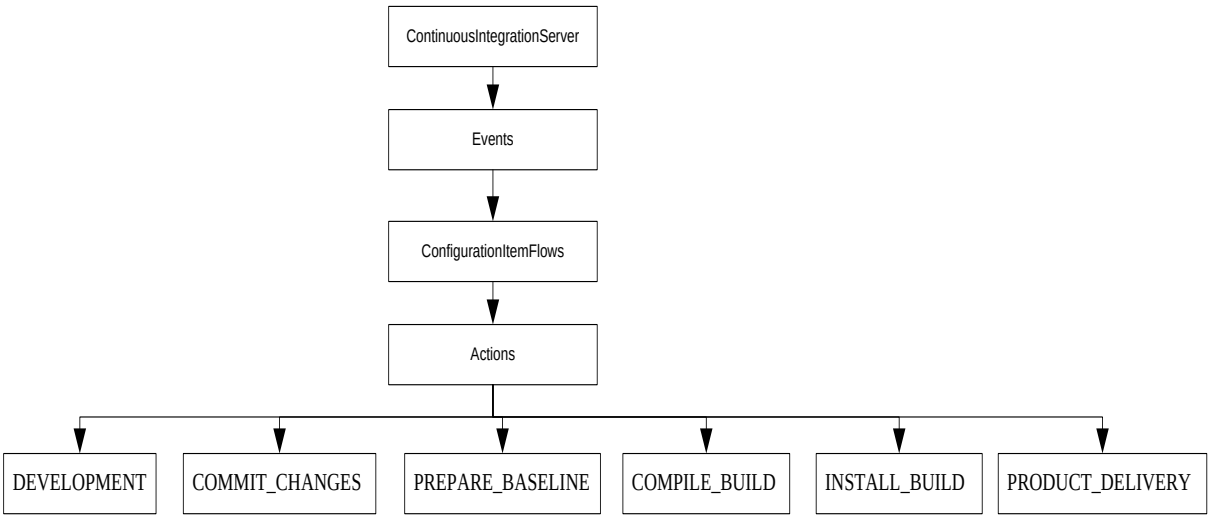


Fig. 3.3. Elements of PIAM meta–model.

Graphical representation of the PIAM model is provided in Fig. 3.4.

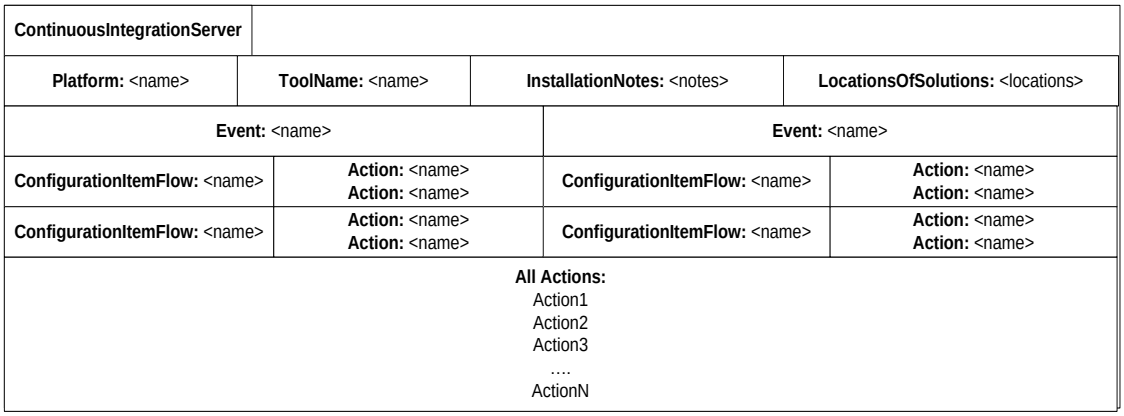


Fig. 3.4. Graphical representation of the PIAM model.

Implementation of Platform Specific Action Model

The aim of platform specific action model (PSAM) is to define the implementation details of a particular platform for all configuration management actions defined by the PIAM. All attributes of all elements, which are empty in the PIAM model, should be fulfilled in the PSAM model. It means that the PSAM model is an extended variant of PIAM, where details about the implementation for a particular platform are given.

The main scope of the PSAM model is the following:

- Storing information about different reusable solutions for each action,
- Managing available solutions for each action for different platforms,
- Adding new reusable solutions for any actions.

To achieve main goals of the PSAM model, a Solution Selecting Module has been designed. The module is represented in Fig. 3.5.

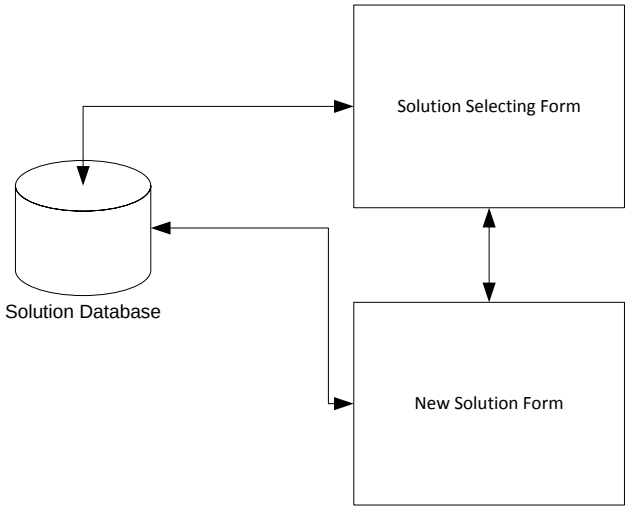


Fig. 3.5. Solution Selecting Module.

Elements represented in Fig. 3.5. have the following description:

- **Solution Database** — stores information about reusable solutions for each action for different platforms.
- **Solution Selecting Form** — allows choosing a particular reusable solution for software configuration management actions.

- **New Solution Form** — allows adding a new solution to a Solution database if the database does not contain the necessary reusable solution for a particular action.

Figure 3.6 represents an entity–relationship diagram for the Solution Database. This ER diagram contains basic requirements for the Solution Database to apply main principles of the PSAM model.

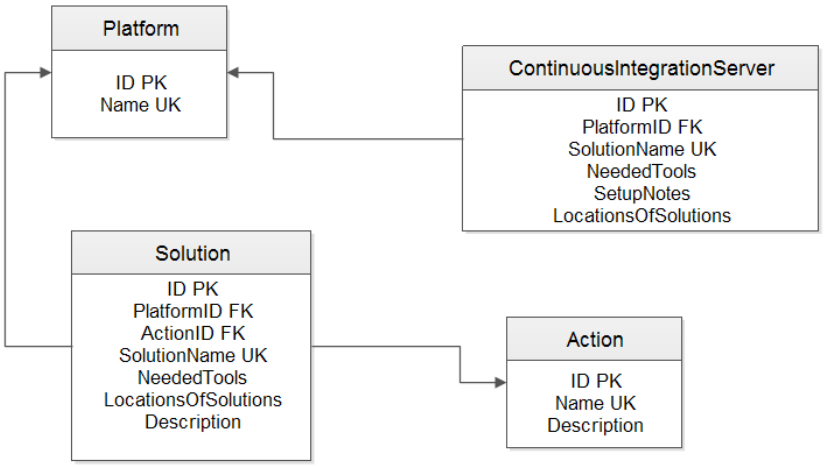


Fig. 3.6. Solution Database.

The algorithm for creation of PSAM model is the following:

1. Getting the PIAM model in the XML format.
2. Parsing element “Actions” and adding each action in the Solution Selecting Form.
3. Software configuration manager works with the Solution Selecting Form. For each action one reusable solution from the Solution Database should be selected and approved by selecting form. Then solutions for all actions are defined, XML file of PIAM model should be fulfilled with details about reusable solution.
4. If software configuration manager detects that some action has not acceptable solution in the database, he should enter a new solution using a New Solution Form. Then a new solution is inserted in the database, the configuration manager returns to step ‘3’.

Figure 3.7 represents an application of basic EAF models for the following environments:

- DEV — a development environment,
- TEST — a test environment,
- Pre_TEST — an environment for testing of builds for a real test environment,
- QA — a quality–accepting environment.

Each original environment has a baseline of source code also represented in Fig. 3.7. Finally, models are implemented using Jenkins continuous integration server.

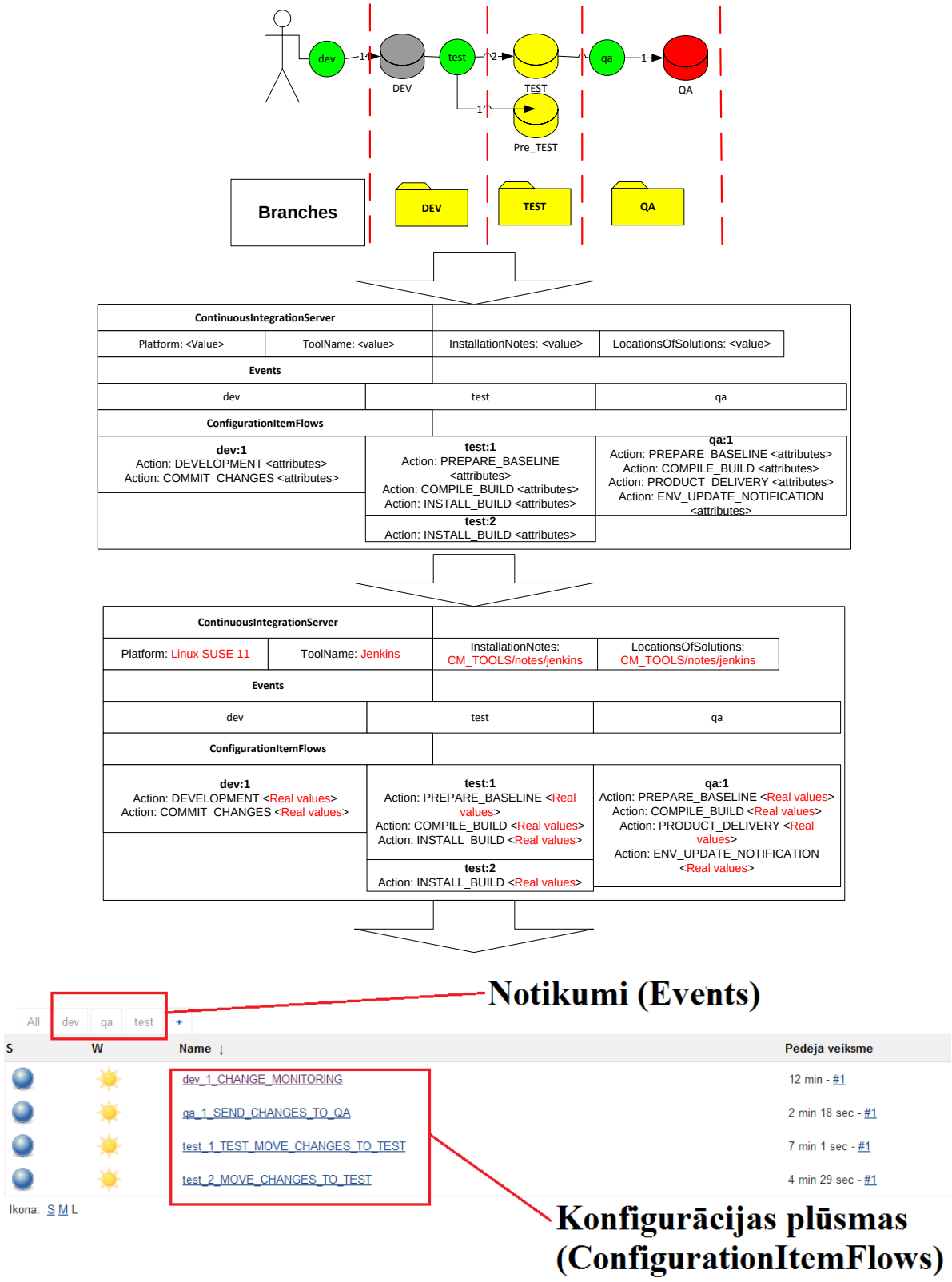


Fig. 3.7. Example of EAF models.

4. APPROBATION AND TESTING OF MODEL–DRIVEN CONFIGURATION MANAGEMENT METHODOLOGY

Preparation for Experiments and Plan

To automate the creation of EAF models, prototype of software has been designed. Tool has been designed and developed by RTU student by individual task of programming. During implementation of tool, the following technologies have been used: .NET, HTML5, CSS un JavaScript, jQuety, KineticJs. Software prototype has been validated by the author of the present Thesis.

The mentioned tool allows creating the following EAF models:

- Environment Model;
- PIAM model;
- PSAM model.

Additionally, the tool supports transformations from EM to PIAM model using transformation rules "E–>P".

For the purpose of experiment, a competence group composed of employees of Tieto Latvia Ltd has been established. The group included senior and leading technical specialists who in their daily work deal with configuration management process automation.

Experiments have the following aims:

- To compare configuration management automation implementation time with that of old methods and EAF methodology.
- To compare the incorrect number of builds before and after the implementation of the EAF methodology.
- Based on comparisons, to determine the changes in the configuration management of automation implementation time, incorrect number of builds in the project, as well as the total number of builds.

Conditions for experiments:

- There is at least one active software development project, which has at least one test environment.
- In the project, the configuration management of software is implemented, the main configuration management tasks described in the first chapter of the Thesis are realized.
- Configuration management process is at least partially automated.

Methods and activities of experiments:

- For experiments five software development and maintenance projects have been selected. In order to increase the reliability of the experiment, projects with different development technologies have been selected.
- For competence group of specialists working at Tieto Latvia Ltd, training courses have been organized, where specialists have been introduced with the offered methodology and models.
- The Solution Choice Module for storing configuration management solutions described in the previous chapter has been developed. Leading programmers have developed software that allows one to manage automation solutions for configuration management tasks. The software consists of the following elements:
  - Oracle database re–used for automation solution storage.
  - Oracle ADF–form that allows one to enter in the database a new configuration management automation solution.

- Oracle ADF–form that receives the ready PIAM model and allows one to choose from the mentioned database an automation solution for each configuration management activity. As a result, the PSAM model is obtained.

- The established Solution Choice Module has been supplemented with solutions during the following steps:
  - Table "ContinuousIntegrationServer" replenishment with solutions for configuration management servers. In this table, information about each configuration management server used in the experiments has been placed.
  - Configuration Manager, who was responsible for each individual experimental project, was able to draw up instructions on how to install the configuration management server, to determine tools, which are necessary in addition, as well as summarize the existing solutions that ease the preparation of configuration management server. When it is done, all the information mentioned above should be placed in the database in table "ContinuousIntegrationServer". The following attributes are filled in the table:
    - **Platform** — a platform, where a specific configuration management server functions;
    - **SolutionName** — a unique name of server;
    - **NeededTools** — a tool list necessary to be implemented in order to activate the configuration management server;
    - **SetupNotes** — detailed instructions of configuration management server installation;
    - **LocationsOfSolutions** — a location of complete solution (if there is one)
  - Filling of table "Solution". Every configuration manager, who is responsible for a particular software project, needs to restructure the corresponding automation solutions in such a way as provided in the PSAM model and the Solution Choice Module**.** Restructuring the existing automation solutions, each of them is placed in the table "Solution" filling in the following table attributes:
    - **Platform** — a platform, in which a given automation solution functions;
    - **Action** — which automates configuration management activities;
    - **SolutionName** —a unique name of automation solution;
    - **NeededTools** — tools that are needed for solution implementation and use;
    - **LocationsOfSolutions** — re–executable location of the code.
    - **Description** — additional instructions for implementation of the solution.
- The evaluation criteria of EAF methodology indicators necessary for the calculation have been developed. For each of five projects from Tieto Latvia Ltd the following data have been obtained:
  - Time spent on the configuration management process for the initial deployment.
  - The average time per week spent on the configuration management process for regular maintenance.
  - Number of weeks until the intended end date of the project.

From each project configuration management database the following information has been obtained:

- The amount of software build,
- The amount of erroneous software build.

In each of the five projects, the implementation of EAF methodology experiment has been carried out according to the following plan:

- The environment model has been created. The model includes the entire development and test environment, as well as the operating environment.
- Environment Model has been transformed into the PIAM model and the SCBM model.

- Configuration Manager works with the Solution Choice Module, supplements configuration management activities in the PIAM model with implementations details. As a result, the PSAM model has been obtained.
- From the PSAM model, the SM model has been obtained, which shows all tools needed to integrate with each other.
- Configuration Manager develops the Service Model (SM) and management system of source code according to the SCBM model.
- PSAM model has been implemented into the configuration management server.
- The time has been fixed, which has been consumed starting with the formation of the Environment Model and ending with the PSAM model implementation.
- Software configuration management process has functioned following the EDF methodology within three months. Subsequently, the following indicators have been fixed:
  - The average time per week necessary to maintain the process and the process of correction of errors.
  - The amount of software build.
  - The amount of erroneous software build.
- The meeting has been held, in which the members of competence group examined time spent and the number of the builds acquired originally and during the experiment.

Evaluation Criteria of Model–Driven Approach

For evaluation of the EAF methodology, the following evaluation criteria have been developed:

- **The time difference in the process of implementation.** Criterion, which shows the percentage difference between time spent on implementation of configuration management automation following the old techniques and the new EAF methodologies. If the value is positive, it means that the implementation of process following the new methodology takes more time than the implementation following old techniques.
- **Time difference of regular maintenance.** Criterion, which shows the percentage difference between time necessary for manual maintenance process before and after the implementation of the EAF methodology. If the value is negative, it means that after the EAF methodology less time is required for manual maintenance process automation. By contrast, in the case of positive values, you need to consume more time for maintenance process after the EDF methodology.
- **Time difference of common maintenance.** A criterion that allows one to judge on the long–term gains of EDF methodology. It takes into account the time for project completion, time necessary for the implementation of the EAF and time necessary for the implementation of the EAF and the time consumed on the average for manual maintenance process before and after the implementation of the EAF. It shows the percentage difference between the common time that would be required to maintain the process until the end of the project following the old techniques and common time that would be required in processes due to the implementation of the EAF.
- **Erroneous build difference.** This criterion shows in percentage changes in the number of erroneous builds in a project after the implementation of the EAF methodology.
- **Common number difference of builds.** This criterion shows in percentage changes in the common number of builds in a project after the implementation of the EAF methodology.

Table 4.1 shows the calculated evaluation criteria of the project.

Table 4.1

The Evaluation Criteria of Project

	Criteria				
Project	**Time difference in process of implementation(%)**	**Time difference of regular maintenance (%)**	**Time difference of common maintenance (%)**	**Erroneous build difference (%)**	**Common number difference of builds (%)**
1	6	–10	23	–27	4
2	–23	–7	36	–40	–43
3	–58	–25	–10	–33	3
4	–11	–38	–28	–60	–3
5	–46	0	71	–29	3

Time difference of automation process

Figure 4.1 shows a schedule, where the horizontal axis contains the project number, and a vertical axis — the time difference of implementation as a percentage.

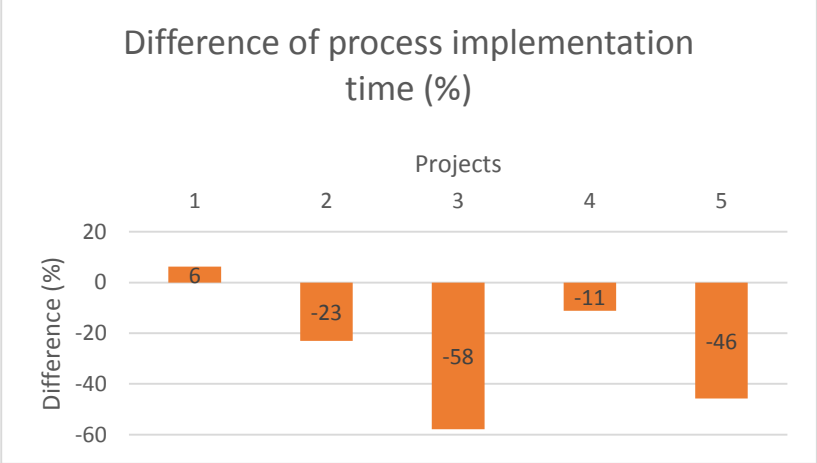


Fig. 4.1. Time difference of automation process implementation comparison by the project.

Analysis of Changes in the Number of Erroneous Builds

Figure 4.2 shows changes in the number of the erroneous builds after implementation of the EAF methodology in all five projects. Schedule reveals another significant benefit of the EAF methodology: the reduction of erroneous build number. The tendency of reduction shows an average reduction of 38 % in the number of erroneous builds.

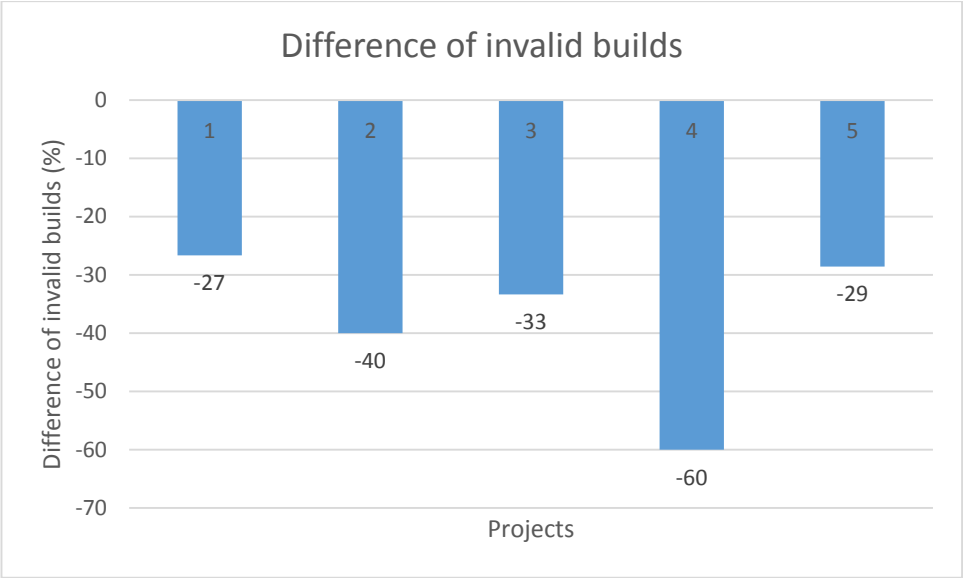


Fig. 4.2. The difference in the number of erroneous bulds.

Conducting experiments mentioned above and analyzing their results, the following EAF methodology benefits have been detected:

- Methodology reduces the time that is required for configuration management process routine maintenance. Due to the fact that all configuration management activities are fulfilled from a centralized place, the level of automation and transparency increases. All activities have an appropriate source code that allows avoiding manual operations. Conducting experiments in five projects, time needed to process the daily maintenance of software has decreased by an average of 16 percent
- The methodology significantly reduces the number of erroneous builds. Creating the source code for each configuration management operation in practice some steps are reviewed, additional quality checks are added, error handling and logging system is improved. This has allowed reducing the number of erroneous builds by an average of 38 percent.
- The implementation of configuration management automation takes less time than after implementation provided by old techniques. The Solution Choice Module contains finished and tested solutions for individual configuration management activity automation. Experiments have shown that if the Solution Choice Module contains implementations for configuration management activities, then the time of process implementation after the EAF methodology is reduced by an average of 34 percent.

Table 4.2 provides the summary of EAF methodology shortages.

Table 4.2

EAF Methodology Shortages

Sequence number of shortage	Description
1	The structure of the Solution Choice Module. As a result of experiment, it has been discovered that it is too extensive to structure a configuration management source code by the main configuration management tasks (compile, deploy, prepare baseline). In this case, functions contain a lot of parameters and functions; body contains a lot of ramifications.
2	The structure of the Environment Model. The existing interpretations of the Environment Model restrict the projects very much. Firstly, in the Environment Model the possibility should be provided that software transmission between environments will take place in several events (Event), and ConfigurationItemFlow can also be subdivided depending on project specifics. Secondly, as noted by the reviewers of conference article and technical specialists, definitions Event and ConfigurationItemFlow are not intuitively understandable. Thus, it would be necessary to find a way to more easily structure configuration management activities, which transmit software changes between the environments. In addition, during environment modeling, the configuration manager should provide the opportunity to more freely structure activities by events and streams. Finally, the definitions of event and flow should be reviewed in order to make definitions intuitively understandable for configuration managers.
3	The essence of PIAM model. A set of actions described in PIAM meta–model, not complete. It should be possible to attach new activities. In addition, transformation from the EM to PIAM model extremely imitates the projects, for which additional steps not defined in the transformation laws should be carried out. When submitting descriptions of models in scientific conferences MODELSWARD 2015, there was a suggestion to combine the EM and PIAM models, allowing the user to choose by himself activities, as well as extend a set of actions in the meta–model.
4	Source code branching model does not reflect a variety of source code management strategies. There are projects that already have other strategies, and in this case the implementation of EAF methodology is impeded by the fact that the methodology requires a certain branching approach. Therefore, there is a suggestion to serve the branching approach only as a recommendation, but to leave some freedom for the branch project name and the branching approach choice.
5	Service model does not provide action with several instances and technologies. Let us suppose that there is a situation, when a particular software release description needs information from a number of different application processing systems. In this case, the Service model has to be flexible enough to allow different systems to connect so that the functions should not take into account the specifics of the project.

The main results of development of the EAF methodology and results of its testing are represented in the following scientific papers: [BAR 2014a, BAR 2014b, BAR 2014c, BAR 2014d, BAR 2014e, BAR 2015].

Justification of the Need for the Second Phase Development of EAF Methodology and Repeated Experiment

The results of experiments have shown that the EAF methodology allows reducing the time of configuration management automation implementation. Introducing configuration management automation to five projects reduces the implementation time by an average of 34% tcompared to introduction of automation by old methods. This tendency on the one hand allows concluding that the aim of this Thesis has been achieved. However, during the experiment, as well as while publishing the EAF methodology foundations in international conference proceedings, essential prerequisites for the second round of development have been reviled:

- Solution Choice Module. When configuration management automation has been introduced in the last of five projects, it has been established that a re–executable exit code becomes difficult to maintain. When discussing the results with specialists of leading competence group, who participated in the organization of experiment, it has been found out that the existing implementation of the EAF methodology is unable to fully provide the repeatedly usable source code for automation processes.
- Reviewer's article reviews [BAR 2015]. It should be noted that this article will be applied for Conference MODEL AWARD 2015 especially dedicated to both MDA and MDD latest achievements. Even though the article has been accepted, one of the reviewers has noted significant deficiencies in the PIAM model, which essentially restricts software configuration managers to create new operations as well as change their order. The reviewer has recommended to combine the Environment Model and the platform independent operation model, so that the user could freely simulate not only the environments but also activities.

Due to the fact that the Solution Development Module, EM and PIAM models are the basic elements of the EAF methodology and these elements need modifications, it has been decided to organize the second development round of the EAF methodology. The main objective has been to improve the structure of solution choice module and to combine the EM and PIAM models.

Taking into account that at the final stage of experiment the EAF methodology basics and the results of experiment have been published in scientific papers, in the methodology it has been necessary to change basic elements. The second round of experiment is necessary, because a number of basic elements of EAF methodology will be changed and, as a result, it will become different. Consequently, it will be necessary to make sure that the first version deficiencies are rectifieda and no benefits are destroyed.

5. IMPROVEMENT OF THE EAF METHODOLOGY

Database Solutions

Database solutions — a method that shows how to keep re–applicable automation solutions for configuration management activities and use these solutions in the EAF models. The method includes the re–used structure of solutions and solution selection algorithm. Figure 5.1 shows the structure of database solutions.

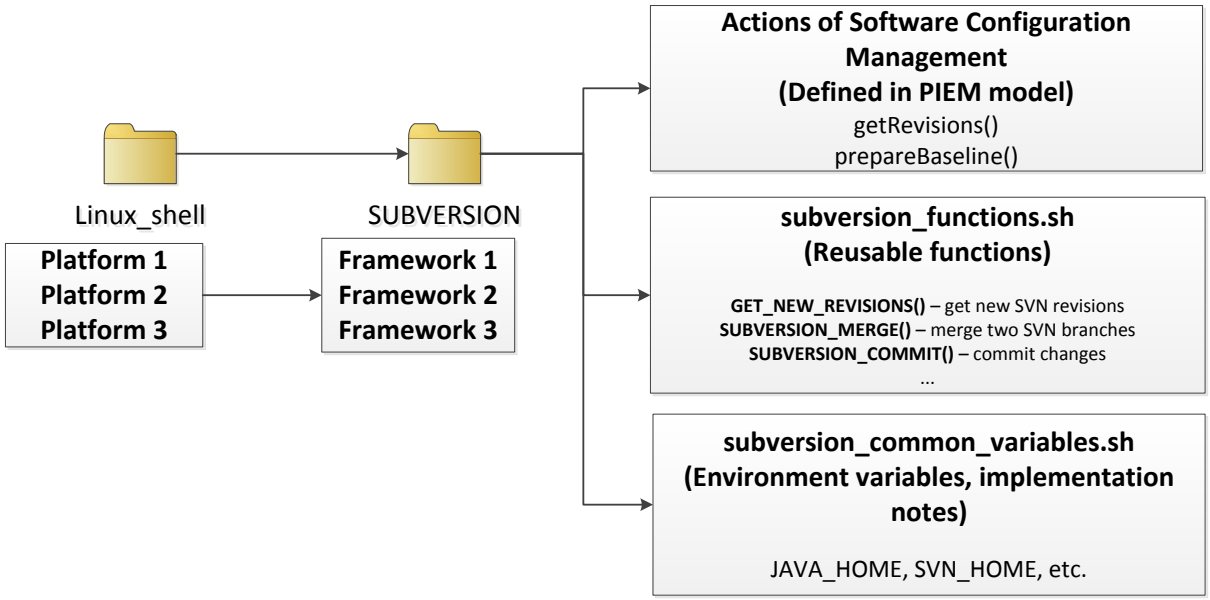


Fig. 5.1. Improved structure of Database solutions.

As can be seen in Fig. 5.1, all re–applicable solutions are grouped into platforms and frameworks. In turn, each frame has the following key attributes:

- Configuration management activities, which are automated with the help of the EAF methodology and defined by the modeling environment project.
- Repeatedly enforceable functions.
- Environment variables and framework implementation guidance.

Working with Database solutions within EAF shown in Fig. 5.1 , the Configuration Manager performs the following steps:

- Selects the platform for all configuration management activities to be implemented. At this point, only frames that match the chosen platform become available to the configuration manager.
- A framework is selected for each operation. At this point, the configuration manager receives the frame functions as a re–used source code for a specific platform and instructions for framework implementations.

Platform Independent Environment Model (PIEM)

This model is the combination of EM and PIAM models defined in the previous chapter, which not only shows the project environment similar to the EM model, but also allows the configuration manager in a moment to define the configuration management action framework. Like the EM model, the new PIEM model does not contain any details of the configuration management activity implementations for a specific platform. Figure 5.2 provides the example of a newly developed model PIEM.

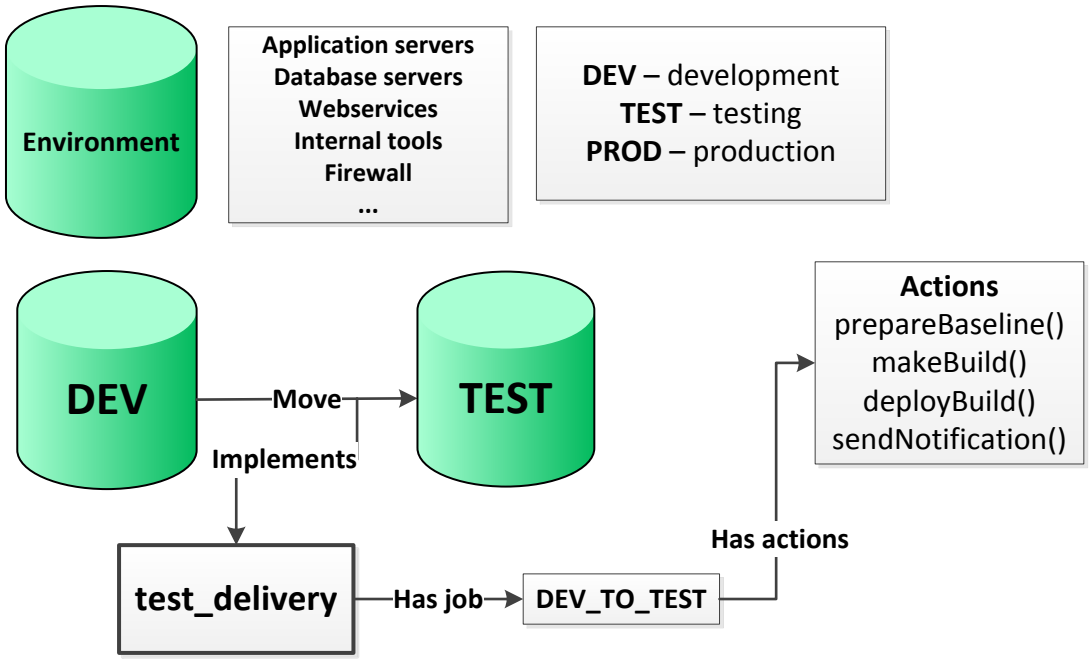


Fig. 5.2. Platform Independent Environment Model.

In the improved version of the EAF methodology, the following improvements have been carried out:

- Modified Solution Database;
- Combined EM and PIAM models by creating PIEM (From platform independent environment model). The main objective was to allow the configuration manager by himself to define the structure of the automated configuration management activities.
- Introduced source model — a directory and file structure, which automatically generates from a platform specific operation model taking into account the specific platform and specific programming language laws.

Figure 5.3 demonstrates the EAF advanced version of model based on the example.

An example that is seen in Fig. 5.3 illustrates the situation, when one software development project has two environments: DEV — development environment and TEST — test environment. It is necessary to regularly transfer software changes from the development to test environment. The process has to be fully automated. It has been decided that automation will be implemented with the help of Jenkins continuous integration server. The server will be installed on the Linux platform and configuration management activities will be automated by Linux shell scripts. EAF task is to generate a source code for the mentioned scripts.

Figure 5.3 shows the EAF models and operational steps:

- Configuration Manager environment, when modeling environment and configuration management activities model the PIEM model;
- PSAM (Platform Specific Operation Model) model is formed from the PIEM. Configuration management operation structure, which is marked in the PSAM model with green, is copied from the PIEM model. In turn, implementation details are granted by the Configuration Manager by selecting for each operation framework from the Solution Database.
- From the PSAM model, the Code Model is generated, which is a set of Linux shell scripts in configuration management operation defined by the PIEM model for automation.

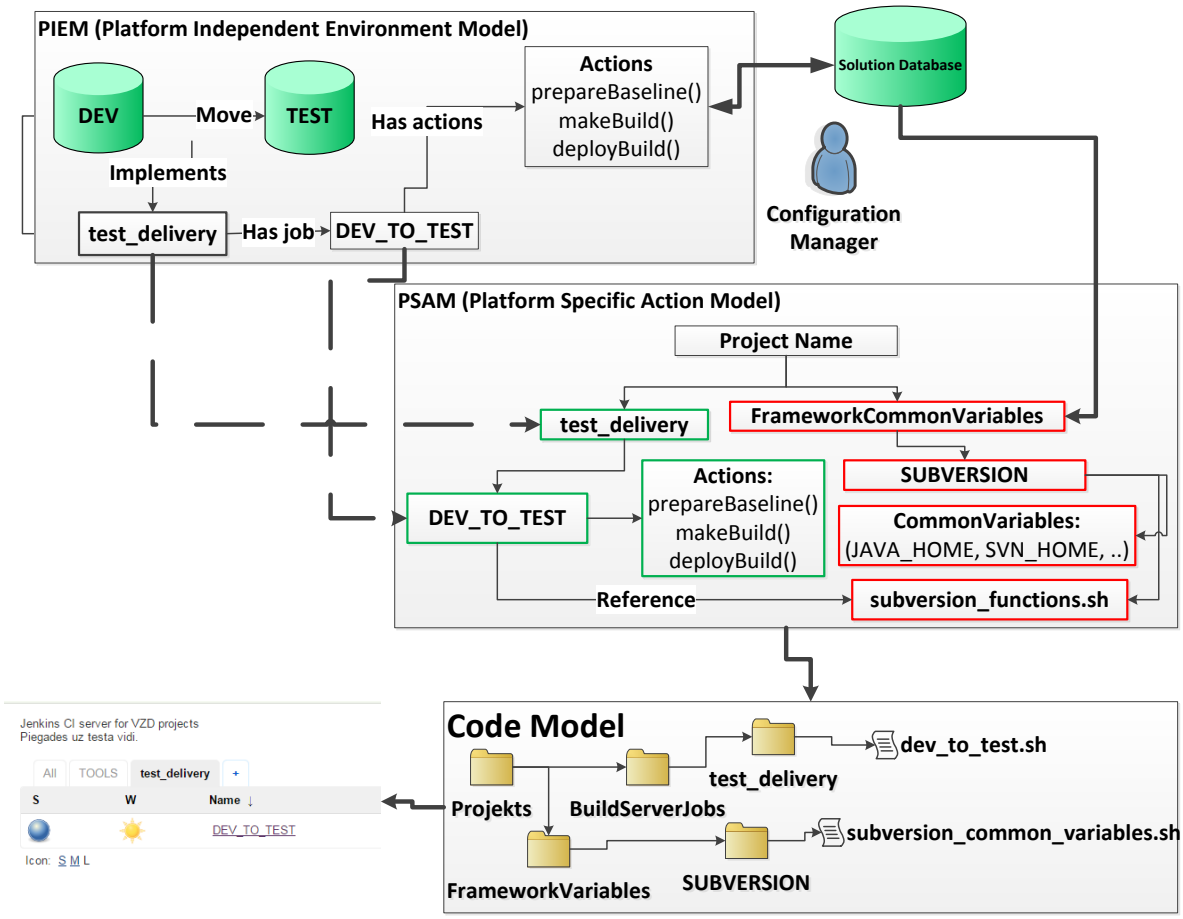


Fig. 5.3. Practical application of improved EAF models.

Testing of Improved Version of the EAF Methodology

When testing a new version of the EAF methodology, experiments have been carried out with the same projects described in the previous chapter. In order to compare the experiment results with the first round of experiments, the same evaluation criteria and the same characteristics have been used. Table 5.1 demonstrates the criteria for the second round of experiments.

Table 5.1

Summary of the Results of Second Round Experiments

Criteria				
Time difference in process of implementation(%)	Time difference of regular maintenance (%)	Time difference of common maintenance (%)	Erroneous builds difference (%)	Common number difference of build (%)
–9	–20	8	–33	–2
–90	–7	–1	–20	–39
–41	–25	–3	–67	–2
–89	–50	–49	–70	3
–96	0	5	–57	2

Thanks to improvements in the EAF methodology, the implementation time of process has been significantly reduced. The average grade of implementing the processes for time reduction in all five projects is 65 percent, which is a very good indicator, considering that initially the Solution Database is empty. Figure 5.1 shows the schedule, in which the time difference between the introduction the first and second experiment round is compared.

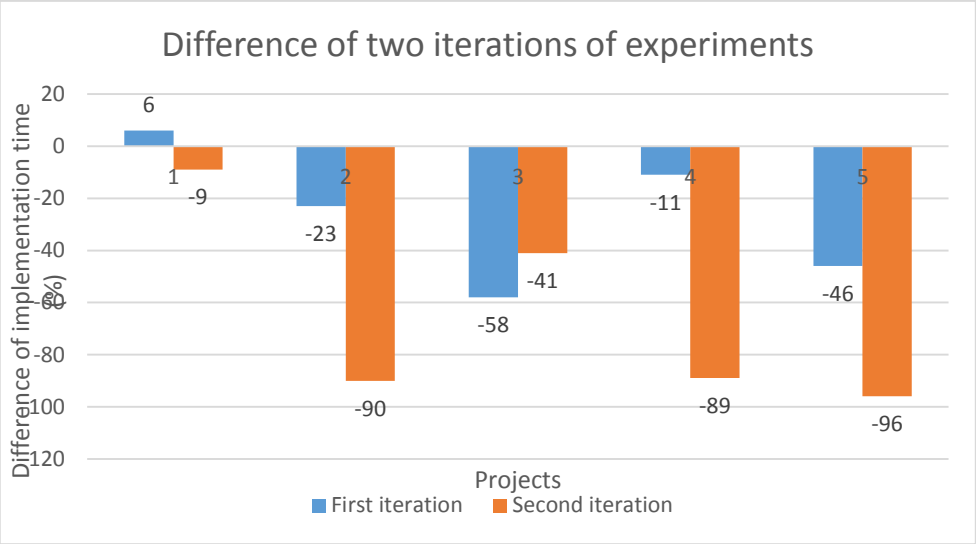


Fig. 5.2. Time comparison of process implementation for two iterations.

Analyzing the results of second round experiments, the following benefits of the EDF methodology have been detected:

- Introducing the EAF methodology in the mentioned five projects, the configuration management automation implementation time decreased by an average of 65 percent. This shows the tendency, that the use of an existing automation solution really allows significantly reducing the introduction of automation into new projects.
- EAF methodology helps to release the project from manual operations in configuration management process maintenance. Thanks to the methodological principles of configuration management executable source code, manual operations are no longer made. Experiment shows that in the project with a relatively low degree of automation, the EAF methodology significantly improves it.

The EAF methodology in experimental projects has reduced the number of erroneous builds by 49 percent. The experiments has shown that thanks to the smart error handling and automated tool mutual integration, the number of erroneous builds in the project decreases. Thanks to the Solution Choice Module complete restructuring the number of erroneous builds is even about 10 % less. This tendency demonstrates storage of the re–used source code for configuration management automation.

The improved version of the EAF methodology and results of experiments of second iteration have been published in the scientific paper [BAR 2015a].

THE MAIN RESEARCH RESULTS, CONCLUSIONS AND FURTHER RESEARCH

The aim of the Thesis has been to develop the Model–Driven approach and methodology for configuration management process automation implementation, which would allow reducing the time of the introduction of automation and improving the quality automation process. To achieve the aim, the following steps have been made:

- The existing solutions and approaches to the automation of configuration management process have been examined.
- The main benefits and disadvantages in the latest configuration management automation solutions have been identified.
- An approach and methodology for automation of configuration management process have been developed.
- A prototype for automation of new methodology implementation has been developed.
- Criteria for new methodology evaluation have been developed.
- Configuration management automation in software development projects have been implemented and following the developed criteria methodology benefits and drawbacks have been defined.
- The improved version of methodology has been developedthat eliminates shortcomings identified as a result of experiment .
- Experiments have been repeated and practical evidence has been gained that the improved version of methodology has removed initially identified deficiencies.

Within this Thesis the methodology has been developed and all new models have experimentally been analyzed in order to verify the hypotheses. Experimental results have shown the following:

- The first hypothesis has demonstrated the comparison of configuration management automation implementation time using the old methods, when existing solutions have not been re–used. Results of experiments have shown the tendency that using the EAF methodology can implement automation approximately twice as fast. In experiments average was 65 %.
- The second hypothesis has proven by comparing with each other configuration management automation timing of introduction after a new methodology for different projects. Experiments have shown that, initially, when the existing solution database is empty and all the solutions have to be constructed from zero, the gain is only 9 percent. In turn, the longer the solutions exist in the database and develop, then they become more stable and automation implementation time decreases. In the last project, in which automation has been implemented, the implementation time decreased by 96 percent compared with the introduction without EAF Model–Driven approach.

While analyzing literature, working with different tools for solving tasks of configuration management and developing a new methodology, it has been concluded:

- Nowadays, the configuration management processes are often incompletely defined, emphasizing only some of the tasks, which are mentioned by industry experts, quality standards and scientific studies.
- Another tendency noticed by the author of this Thesis is that the configuration management tends to be regarded simply as a set of tools. Sometimes the industry professionals believe that installing the tools, processes can be considered introduced and they no longer have to worry about them.
- This position has led to losses not only for projects in Latvia but also around the world. No matter what tools are used in the project, it is important to choose the effective tool

implementing methodology, which could effectively choose, configure tools, as well as make recommendations performing configuration management process activities.

Results of the research have been used in research projects and in one RTU Course "Applied Computer Software" in the study process:

- The execution of the project by the Latvian Council of Science "The Development of Models and Methods for Constructive Intellectual Software Based on Dispersive Artificial Intellect, Knowledge Management and Progressive Web Technologies" (Prof. J. Grundspenkis); The development of Model–Driven software management methods.
- The European Commission's 7th FP project eINTERASIA "ICT Transfer Concept for Adaptation, Dissementation and Local Exploitation of European Research Results in Central Asia's Countries", 2013–2015 (project coordinator — prof. Leonid Novitsky); The development of manegements models in software framework.
- The study course "Applied Computer Software". Learning tool has been developed (subject "Applied Computer Software" / L. Novitsky, V. Kotov, A. Lesovskis A. Bartusevičs, RTU, 2012. – 67 p.; Section: Applied Software Configuration Management).
- National research project VVP Y8089 "Cyber Physical Systems, Ontologies and Biofotonita for Safe and Easy City and the Public (since 2014) — Software Configuration Management".

Further development directions of the Doctoral Thesis:

- Environment initial installation process formalization. Currently, the EAF methodology requires that all environments have already been established. However, actually, at the very beginning of software development project environments are created from zero: the operating system, application servers, databases, configures firewalls and so on are installed.
- The EAF methodology compliance should be accessed for the most popular quality standards and guidelines, such as CMMI, ISO, ITIL, etc.
- One more model should be developed that would allow automatically generating a configuration management plan from the PEM, PSAM and CM models.

BIBLIOGRAPHY

[ABO 2014] About CMMI Institute. 2014. [ONLINE] Available at: http://whatis.cmmiinstitute.com/about–cmmi–institute. [Apskatīts 02 Septembrī 2014].

[AIE 2010] Aiello, R. Configuration Management Best Practices: Practical Methods that Work in the Real World (1st ed.). Addison–Wesley, 2010.

[ALT 2008] Altmanninger K. Models in conflict – towards a semantically enhanced version control system for models. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2008;5002 LNCS:293–304.

[ALT 2010] Bruce Altner, Brett Lewinski. A Roadmap to Continuous Integration. Proceedings of the 2010 IT Summit, NASA, 2010.

[ASN 2010] Asnina, E. & Osis, J. 2010, "Computation independent models: Bridging problem and solution domains", Proceedings of the 2nd International Workshop on Model–Driven Architecture and Modelling Theory–Driven

Development, MDA and MTDD 2010, in Conjunction with ENASE 2010, pp. 23.

[AZO 2008] Azoff M., The Benefits of Model Driven Development. MDD in Modern Web–based Systems, Published March, Butler Direct Limited, 2008.

[AZO 2014] Azoff, R., DevOps: Advances in Release Management and Automation. [ONLINE] Available at: http://electric–cloud.com/wp–content/uploads/2014/06/EC–IAR_Ovum–DevOps.pdf [Apskatīts 20 Oktobrī 2014].

[BAM 1995] Bamford, R., 1995. *Configuration Management and ISO 9001*. Software Systems Quality Consulting, DO–25 V6, 7., 1995.

[BAR 2012a] Bartusevics A., Kotovs V., Novickis L. A Method for Effective Reuse–Oriented Software Release Configuration and Its Application in Insurance Area. Proceedings of Riga Technical University „Information Tehnology and Management Science", 15th series, RTU Publishing, 2012, Riga, Latvia, pp. 111–115.

[BAR 2012b] Bartusevics A., Kotovs V. Towards the effective reuse–oriented release configuration process. Proceedings of the 5–th International Scientific Conference „Applied Information and Communication Tehnologies", 2012, Jelgava, Latvia, pp. 99–103.

[BAR 2013] Bartusevics A., AMethodology for Model–Driven Software Configuration Management Implementation and Support. Proceedings of the 6th International Scientific Conference „Applied Information and Communication Tehnologies", 2013, Jelgava, Latvia, pp. 252–258.

[BAR 2014a] Bartusevičs, A., Novickis, L. Model–Driven Software Configuration management and Environment Model. No: Recent Advances in Electrical and Electronic Engineering. Proceedings of the 3rd International Conference on Systems, Communications, Computers and Applications (CSCCA"14), Itālija, Florence, 22.–24. novembris, 2014. Italy: WSEAS Press, 2014, 132.–140. lpp. ISBN 978–960–474–399–5. ISSN 1790–5117.

[BAR 2014b] Bartusevičs, A., Novickis, L., Leye, S. Implementation of Software Configuration Management Process by Models: Practical Experiments and Learned Lessons. Applied Computer Systems. Nr.16, 2014, 26.–32. lpp. ISSN 2255–8683. e–ISSN 2255–8691. Pieejams: doi:10.1515/acss–2014–0010

[BAR 2014c] Bartusevičs, A., Novickis, L. Models for Implementation of Software Configuration Management. No: Procedia Computer Science. Valmiera, Latvia: 2014, 3.–10. lpp.

[BAR 2014d] Bartusevičs, A., Lesovskis, A., Novickis, L. Model–Driven Software Configuration Management and Semantic Web in Applied Software Development. Proceedings of the 13th International Conference on Telecommunications and Informatics (TELE–INFO '14), IIstanbul, Turkey December 15–17, 2014.

[BAR 2014e] Bartusevičs, A., Novickis, L. Towards the Model–driven Software Configuration Management Process. Information Technology and Management Science. Nr.17, 2014, 32.–38. lpp. ISSN 2255–9086. e–ISSN 2255–9094.

[BAR 2014f] Bartusevics Arturs, Leonids Novickis and Eberhard Bluemel. 2014. Intellectual Model–Based Configuration Management Conception. Applied Computer Systems. 15(1): 5–41. Retrieved 28 Nov. 2014, from doi:10.2478/acss–2014–0003

[BAR 2015] Bartusevičs, A., Novickis, L. Model–based Approach for Implementation of Software Configuration Management Process. No: MODELSWARD 2015: Proceedings of the 3rd International Conference on Model–Driven Engineering and Software Development, Francija, Angers, 9.–11. februāris, 2015. Lisbon: SciTePress, 2015, 177.–184. lpp. ISBN 978–989–758–083–3.

[BAR 2015a] Bartusevičs, A., Lesovskis, A., Novickis, L. Semantic Web Technologies and Model–Driven Approach for the Development and Configuration Management of Intelligent Web–Based Systems. No: Proceedings of the 2015 International Conference on Circuits, Systems, Signal Processing, Communications and Computers, Austrija, Vienna, 15.–17. marts, 2015. Vienna: 2015, 32.–39. lpp. ISBN 978–1–61804–285–9. ISSN 1790–5117.

[BEL 2005] Bellagio, M. What Is Software Configuration Management? Internet http://ptgmedia.pearsoncmg.com/images/0321200195/samplechapter/bellagio_ch01.pdf, 2005.

[BER 2003] Berczuk, Appleton. Software Configuration Management Patterns: Effective TeamWork, Practical Integration (1st ed.). Addison–Wesley, 2003.

[BER 2011] Berziša, S. & Grabis, J. 2011, "Combining project requirements and knowledge in configuration of project management information systems", ACM International Conference Proceeding Series, pp. 89.

[BER 2012] Bērziša, Solvita. Application of Knowledge and Best Practices in Configuration of Project Management Information Systems : promocijas darbs / S.Bērziša ; zinātniskais vadītājs J.Grabis ; Rīgas Tehniskā universitāte. DATORZINĀTNES UN INFORMĀCIJAS TEHNOLOĢIJAS FAKULTĀTE. Informācijas tehnoloģijas institūts. Vadības informācijas tehnoloģijas katedra. Rīga : [RTU], 2012. 196 pp.

[BIL 2014] Bill Chamberlin's HorizonWatching. 2014. Top 18 Trends in Application Software Development for 2014. [ONLINE] Available at: http://www.billchamberlin.com/top–18–trends–in–application–software–development–for–2014/. [Apskatīts 20 Oktobrī 2014].

[BRA 2008] Bastian Braun. SAVE – Static Analysis on Versioning Entities. ICSE: International Conference on Software Engineering, 2008.

[BRO 2002] Brouse, Peggy S. Configuration Management Interenet: http://www.eolss.net/Sample–Chapters/C15/E1–28–03–02.pdf , 2002.

[BRO 2005] A. B. Brown, A. Keller, and J. L. Hellerstein, A model of configuration complexity and its application to a change management system, IFIP/IEEE International Symposium, Integrated Network Management, pp. 631– 644, 2005.

[BRU 2004] Brugge, B., Dutoit, A. Software Configuration Management. Internet https://files.ifi.uzh.ch/rerg/amadeus/teaching/courses/software_engineering_hs08/folien/Kapitel_23_Addendum_SCM.pdf, 2004.

[BUC 2009] Buchmann T., Dotor A., Westfechtel B. MODEL–DRIVEN DEVELOPMENT OF SOFTWARE CONFIGURATION MANAGEMENT SYSTEMS. ICSOFT 2009 – 4th International Conference on Software and Data Technologies 2009.

[BUS 2011] Bushehrian O., Automatic object deployment for software performance enhancement. The Institution of Engineering and Technology 2011, Vol. 5, Iss. 4, pp. 375–384, 2011.

[CAL 2012] Calhau R., Falbo R. A Configuration Management Task Ontology for Semantic Integration. Proceedings of the 27th Annual ACM Symposium on Applied Computing Pages 348–353 ACM New York, NY, USA, 2012.

[CLE 2012] Clemencic M., Mato P., A CMake–based build and configuration framework. Journal of Physics: Conference Series 396 (2012) 052021, 2012.

[CMC 2014] CMCrossroads | Three Major Trends in Software Release Management You Should Adopt . 2014. [ONLINE] Available at: http://www.cmcrossroads.com/article/three–major–trends–software–release–management–you–should–adopt. [Apskatīts 20 Oktobrī 2014].

[COM 2011] Comas J., Mostashari A., Mansouri M., Turner R. A Software Deployment Risk Assessment Heuristic for Use in a Rapidly–Changing Business–to–Consumer Web Environment International Journal of Software Engineering and Its Applications Vol. 5 No. 4, October, 2011.

[CON 2002] Configuration Management Training. Section 1: Explaining Configuration Management, EESA, 2002. Internet: http://esamultimedia.esa.int/docs/industry/SME/Configuration/Section_1–CM.pdf

[CON 2015] The Convergence of DevOps « IT Revolution IT Revolution. [ONLINE] Available at: http://itrevolution.com/the–convergence–of–devops/. [Apskatīts 28 Janvārī 2015].

[CRA 2008] Cravino P., Enterprise Software Configuration Management Solutions for Distributed and System z. 1st ed. USA: Redbooks. 2008.

[DAR 2001] Dart, S. Concepts in Configuration Management Systems. Internet http://sceweb.uhcl.edu/boetticher/swen5230/concepts–in–configuration–management.pdf, 2001.

[DEP 2010] Department of Defense, USA Military Handbook. Configuration management guidance (rev. A) (MIL–HDBK–61A). Retrieved January 5, 2010, from http://www.everyspec.com/MIL–HDBK/MIL–HDBK–0001–0099/MIL–HDBK–61_11531/

[DEV 2014] DevOps Implementation | Giga Promoters. 2014. [ONLINE] Available at: http://gigapromoters.com/offerings/services/it–services/devops–implementation/. [Apskatīts 10 Novembrī 2014].

[DOD 2014] Do DevOps tools really exist?. 2014. Do DevOps tools really exist?. [ONLINE] Available at: http://www.scriptrock.com/blog/devops–tools–exist. [Apskatīts 11 Novembrī 2014].

[DON 2011] Doniņš Uldis. Topoloģiskā biznesa sistēmu modelēšana un programmatūras sistēmu projektēšana. Metodiskais līdzeklis. RTU Izdevniecība. Rīga 2011.

[EIL 2006] T. Eilam, M. H. Kalantar, A. V. Konstantinou, G. Pacifici, and J. Pershing, "Managing the Configuration Complexity of Distributed Applications in Internet Data Centers," IEEE Communications Magazine, vol. 44, pp. 166–177, 2006.

[EST 2013] Estler, H.–Christian, Unifying Configuration Management with Merge Conflict Detection and Awareness Systems. In 22nd Australian Conference on Software Engineering. Austalia, 4–7 June 2013. Austalia: IEEE. 201–210.

[FIT 2014] Fitzgerald B., Stol J., Continuous software engineering and beyond: trends and challenges. Proceeding in RCoSE 2014 Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering, Pages 1–9. ACM New York, NY, USA, 2014.

[FUG 2014] Fuggetta A., Nitto E., Software process. Proceeding in FOSE 2014 Proceedings of the on Future of Software Engineering, Pages 1–12, ACM New York, NY, USA, 2014.

[GAL 2009] Galup, S. D., Dattero, R., Quan, J.J., Conger, S., An Overview of IT Service Management. *Commun.* ACM, 2009, vol. 52, no. 5, pp. 124–127., 2009.

[GHE 2012] Giacomo Ghezzi, Michael Würsch, Emanuel Giger, Harald Gall. An Architectural Blueprint for a Pluggable Version Control System for Software (Evolution) Analysis, In: 2nd Workshop on Developing Tools as Plug–ins, Zurich, 03 June 2012 – 03 June 2012.

[GIE 2009] Giese Holger, Seibel Andreas, Vogel Thomas. A Model–Driven Configuration Management System for Advanced IT Service Management. Available at: http://www.hpi.unipotsdam.de/giese/gforge/publications/pdf/GSV–MRT09_paper_7.pdf, 2009.

[GLO 2012] IT Glossary. Defining The IT Industry. SCM Software Configuration Management. Internet http://www.gartner.com/it–glossary/scm–software–configuration–management/, 2012.

[GRO 2007] H. Gronniger, H. Krahn, B. Rumpe, M. Schindler, and S. Volkel. Textbased Modeling. In 4th International Workshop on Software Language Engineering, 2007.

[GUO 2005] Guozheng Ge, E., Whitehead, Jr. Automatic Generation of Rule–based Software Configuration Management Systems. ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA.

[HAG 2010] Hagen, S., Kemper, A., Model–Based Planning for State–Related Changes to Infrastructure and Software as a Service Instances in Large Data Centers. Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, On page(s): 11–18, Volume: Issue: , 5–10 July 2010

[HAT 2012] Hideaki Hata, Osamu Mizuno, Tohru Kikuno. Bug Prediction Based on Fine–Grained Module Histories. ICSE: International Conference on Software Engineering, Feb2012, p200–210.

[HIS 2014] History of software configuration management – Wikipedia, the free encyclopedia. 2014. [ONLINE] Available at: http://en.wikipedia.org/wiki/History_of_software_configuration_management. [Apskatīts 05 Novembrī 2014].

[HIST 2014] A History of Version Control. [ONLINE] Available at: http://ericsink.com/vcbe/html/history_of_version_control.html. [Apskatīts 11 Novembrī 2014].

[HUA 2009] Shi–Ming Huang, Chih–Fong Tsai, Po–Chun Huang. Component–based software version management based on a Component–Interface Dependency Matrix, The Journal of Systems and Software, 2009.

[JIA 2009] Jiang, L., Eberlein, A., An Analysis of the History of Classical Software Development and Agile Development. Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA – October 2009.

[JOH 2011] Johnsen E., Schlatte R. Integrating Aspects of Software Deployment in High–Level Executable Models, presented at the NIK–2011 conference, 2011.

[JUI 2002] Juite Wanga, Yung–I Lin, A fuzzy multicriteria group decision making approach to select configuration items for software development. MathematicsWEB, Fuzzy Sets and Systems, 2002.

[KAN 2005] Ronald Kirk Kandt. Configuration Management Principles and Practices. Jet Propulsion Laboratory, 4800 Oak Grove Dr., Pasadena, CA 91 109, USA Internet: http://trs–new.jpl.nasa.gov/dspace/bitstream/2014/10507/1/02–2525.pdf , 2005.

[KAP 2008] Kapitza R, Baumann P, Reiser HP. Using object replication for building a dependable version control system. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2008;5053 LNCS:86–99.

[KAR 2009] G. Karsai, H. Krahn, C. Pinkernell. Design Guidelines for Domain Specific Languages. Proceedings of the 9th OOPSLA Workshop on Domain–Specific Modeling DSM'09, page 7–13., 2009.

[KEL 2008] S. Kelly and J.–P. Tolvanen. Domain–Specific Modeling: Enabling Full Code Generation. Wiley, 2008.

[KR 2014] Krusche S., Alperowitz L., Introduction of continuous delivery in multi–customer project courses. Proceeding in ICSE Companion 2014 Companion Proceedings of the 36th International Conference on Software Engineering, Pages 335–343. ACM New York, NY, USA, 2014.

[LAV 2011] Jannik Laval, Simon Denier, Stéphane Ducasse, Jean–Rémy Falleri. Supporting simultaneous versions for software evolution assessment. Science of Computer Programming, 2011.

[LES 2014] Lesson 11: Devops & Configuration Management Intro — OSU DevOps Bootcamp 0.0.1 documentation. 2014. [ONLINE] Available at: http://devopsbootcamp.readthedocs.org/en/latest/11_devops.html. [Apskatīts 10 Novembrī 2014].

[LI 2012] Jingyue Li, Michael D. Ernst. CBCD: Cloned Buggy Code Detector. ICSE: International Conference on Software Engineering, Feb2012, p310–320.

[MAL 2012] Malek S. An Extensible Framework for Improving a Distributed Software System's Deployment Architecture. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 38, NO. 1, JANUARY/FEBRUARY 2012.

[MEL 2006] Mellon, K. A Framework for Software Product Line Practice, Version 5.0. Internet: http://www.sei.cmu.edu/productlines/frame_report/config.man.htm, 2006.

[MER 2005] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain–specific languages. Technical Report SEN–E0309, Centrum voor Wiskunde en Informatica, Amsterdam, 2005.

[MET 2002] Anne Mette Jonassen Hass. Configuration Management Principles and Practice, Addison–Wesley Professional. Part of the Agile Software Development Series series, 2002, pages 432.

[MUR 2008] Leonardo Murta, Chessman Correa, Joao Gustavo Prudencio. Towards Odyssey–VCS 2: Improvements over a UML–based Version Control System. ICSE: International Conference on Software Engineering, 2008.

[NIK 2008] Nikulsins, V. & Nikiforova, O. 2008, "Adapting software development process towards the model driven architecture", Proceedings – The 3rd International Conference on Software Engineering Advances, ICSEA 2008, Includes ENTISY 2008: International Workshop on Enterprise Information Systems, pp. 394.

[NIK 2009] Nikiforova, O., Cernickins, A. & Pavlova, N. 2009, "Discussing the difference between model driven architecture and model driven development in the context of supporting tools the projection of two–

hemisphere model into the component model of MDA/MDD", 4th International Conference on Software Engineering Advances, ICSEA 2009, Includes SEDES 2009: Simposio para Estudantes de Doutoramento em Engenharia de Software, pp. 446.

[OPE 2014] OpenMake Products. [ONLINE] Available at: http://www.openmakesoftware.com/build–management. [Apskatīts 22 Novembrī 2014].

[OSE 2002] Oject–Oriented Software Engineering Using UML, Patters and JAVA "Software Configuration Management" Internet: http://www.bilkent.edu.tr/~bakporay/cs_413/Bruegge_L28_Configuration Management_ch12lect1.ppt , 2002

[OSI 2008] Osis, J., Asnina, E. & Grave, A. 2008, Formal problem domain modeling within MDA. Proceedings of the 2nd International Conference on Software and Data Technologies, ICSOFT 2007; Barcelona; Spain; Volume 22 CCIS, 2008, Pages 387–398.

[OSI 2010] Osis, J. & Donins, U. 2010, "Platform independent model development by means of topological class diagrams", Proceedings of the 2nd International Workshop on Model–Driven Architecture and Modelling Theory–Driven Development, MDA and MTDD 2010, in Conjunction with ENASE 2010, pp. 13.

[OSI 2011] Osis J., Asnina E. Model–Driven Domain Analysis and Software Development: Architectures and Functions. IGI Global, Hershey – New York, 2011, 514 p.

[PAI 1999] R. Paige, J. Ostroff, and P. Brooke. Principles for Modeling Language Design. Technical Report CS–1999–08, York University, December 1999.

[PAU 2007] Paul M. Duvall, Steve Matyas, and Andrew Glover. Continuous Integration: Improving Software Quality and Reducing Risk. (1st ed.). Addison–Wesley Professional, 2007.

[PFA 1997] P. Pfahler and U. Kastens. Language Design and Implementation by Selection. In Proc. 1st ACM–SIGPLAN Workshop on Domain–Specific–Languages, DSL '97, pages 97–108, Paris, France, January 1997. Technical Report, University of Illinois at Urbana–Champaign.

[PIN 2009] Pindhofer Walter, Model Driven Configuration Management. Master work of Wien University, Wien, 2009.

[RAG 2014] Ragan, T., 21st–Century DevOps—an End to the 20th–Century Practice of Writing Static Build and Deploy Scripts, Linux Journal, 230, pp. 116–120, Computers & Applied Sciences Complete, EBSCOhost, viewed 22 October 2014.

[RAZ 2007] Saad Razzaq, Fahad Maqbool, Bilal Anjum. The Challenges & Case for Mining Software Repositories. International MultiConference of Engineers and Computer Scientists, 2007.

[ROS 2010] Alessandro Rossini, Adrian Rutle, Yngve Lamo, UweWolter. A formalisation of the copy–modify–merge approach to version control in MDE. The Journal of Logic and Algebraic Programming, 2010.

[RUA 2003] Ruan Li, Zhong Yong, A New Configuration Management Model for Software Based on Distributed Components and Layered Architecture. Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. 27–29 Aug. 2003.

[SAR 2008] Anita Sarma, David Redmiles, André van der Hoek. Empirical evidence of the benefits of workspace awareness in software configuration management.

Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, 2008

[SAT 2011] Laika Satish, Identifying the Dissimilarities based on Working of Programs among Versions in DVCS (Distributed Version Control Systems). International Journal of Computer Applications (0975 – 8887) Volume 36– No.6, December 2011.

[SCH 2010] Holger Schackmann, Horst Lichter. Process assessment by evaluating configuration and change request management systems. Proceedings of the Warm Up Workshop for ACM/IEEE ICSE 2010

[SCM 2001] Software Configuration Management Internet: http://dogbert.mse.cs.cmu.edu/charlatans/References/Configuration%20Management/0130912972.pdf 2001.

[SEK 2012] Atsuji Sekiguchi, Kuniaki Shimada, Yuji Wada, Akio Ooba, Ryouji Yoshimi, Akiko Matsumoto. Configuration management technology using tree structures of ICT systems. Proceedings of the 15th Communications and Networking Simulation Symposium Publisher: Society for Computer Simulation International, 2012

[SER 2014] Serena Deployment Automation Overview. Serena Deployment Automation Overview. [ONLINE] Available at: http://www.serena.com/index.php/en/products/featured–products/serena–deployment–automation/overview/. [Apskatīts 22 Novembrī 2014].

[SHI 2010] Shih C., Huang S. Exploring the relationship between organizational culture and software process improvement deployment, Information & Management 47 (2010) 271–281p., 2010.

[SIN 2008] Sindhuja P. N., Surajit Ghosh Dastidar. Software Deployment: Concepts and Technologies. ICFAI Journal of Systems Management, 2008.

[SIN 2010] Sinan Si Alhir. Understanding the Model Driven Architecture (MDA). Available at: http://www.methodsandtools.com/archive/archive.php?id=5, 2010.

[SIY 2008] Harvey Siy, Parvathi Chundi, Daniel J. Rosenkrantz, Mahadevan Subramaniam. A segmentation–based approach for temporal analysis of software version repositories. Jornal of Software Maintenance and Evolution: Research and Practice, 2008.

[SOF 2014] Software configuration management – Wikipedia, the free encyclopedia. 2014. [ONLINE] Available at: http://en.wikipedia.org/wiki/Software_configuration_management. [Apskatīts 05 Novembrī 2014].

[STA 2008] Glen Stansberry. 7 Version Control Systems Reviewed. At http://www.smashingmagazine.com/2008/09/18/the–top–7–open–source–version–control–systems/, 2008.

[TAK 2014] Taking Release Management to the Next Level. 2014. [ONLINE] Available at: http://www.slideshare.net/xebialabs/taking–releasemanagementtothenextlevel. [Apskatīts 20 Oktobrī 2014].

[TAR 2011] Alexander Tarvo, Thomas Zimmermann, Jacek Czerwonka. An integration resolution algorithm for mining multiple branches in version control systems. IEEE international conference on software maintenance, ICSM; 2011. 402 p.

[THA 2009] Tung Thanh Nguyen, Hoan Anh Nguyen, Nam H. Pham, Jafar M. Al–Kofahi, Tien N. Nguyen. Clone–Aware Configuration Management. ASE

'09: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, 2009.

[TOL 2005] J. Tolvanen, S Kelly. Defining Domain–Specific Modeling Languages to Automate Product Derivation: Collected Experiences. Proceeding SPLC'05 Proceedings of the 9th international conference on Software Product Lines Pages 198–209. 2005.

[TRE 2014] Trends in Software Engineering – Dice News. 2014. [ONLINE] Available at: http://news.dice.com/software–engineering–talent–community/trends/. [Apskatīts 20 Oktobrī 2014].

[VAC 2006] VACCAPERNA Systems Limited. Software Configuration Management (SCM). Internet http://www.vaccaperna.co.uk/scm/about_scm.html, 2006.

[VAS 2013] Vasiljevics, I., Milosavljevics, G., Dejanovics, I., Filipovics, M., COMPARISON OF GRAFICAL DSL EDITORS. The 6th PSU–UNS International Conference on Engineering and Technology (ICET–2013), Novi Sad, Serbia, May 15–17, 2013.

[WET 2012] Wettinger J., Concepts for Integrating DevOps Methodologies with Model–Driven Cloud Management Based on TOSCA. Institute of Architecture of Application Systems University of Stuttgart, 2012.

[WHA 2014] What Are Current Hot Trends In The Field Of Software Engineering?. 2014. [ONLINE] Available at: http://bloggless.com/it/software–engineering/what–is–currently–popular–in–software–engineering/. [Apskatīts 20 Oktobrī 2014].

[WIL 2003] D. Wile. Lessons Learned from Real DSL Experiments. Proceedings of the 36th Hawaii International Conference on System Sciences, 2003.

[WIL 2004] D. Wile. Lessons learned from real DSL experiments. Science of Computer Programming, 51(3):265–290, June 2004.

[WES 2005] Westfechtel, B., Conradi, R. Software Architecture and Software Configuration Management Internet http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.3085&rep=rep1&type=pdf, 2005.

[ЗАМ 2008] Заметки о Software Configuration Management. Управление конфигурацией программного обеспечения. Интернет: http://scm–notes.blogspot.com/p/scm–books.html, 2008.

[ЛАП 2004] Лапыгин, Д. Новичков, А. Конфигурационное управление проектами разработки программного обеспечения. Интернет: http://citforum.ru/SE/quality/configuration_management/, 2004.

[ОРЛ 2011] Орлик, С. Программная инженерия. Конфигурационное управление Перевод главы из SWEBOK с комментариями. Архивировано из первоисточника 14 марта 2012. Проверено 18 июня 2011.

[УДО 2011] Удовиченко, Ю. Управление изменениями и кессонная болезнь проектов. Интернет: http://experience.openquality.ru/software–configuration–management/, 2011.