Model-based Approach for Implementation of Software Configuration Management Process

Arturs Bartusevics and Leonids Novickis
Faculty of Computer Science and Information Technology, RTU, Kalku Street 1, Riga, Latvia

Keywords: Software Configuration Management, Model-Driven Approach, Models.

Abstract: Software configuration management is a discipline that controls software evolution process. Nowadays this process is not only challenge to choose the best version control system o branching strategy for particular project. Together with source code management the following tasks should be solved: continuous integration, continuous delivery, release management, build management etc. Usually software development companies already have a set of tools to support mentioned processes. The main challenge is to adopt this solutions to new projects as soon as possible with minimum efforts of manual steps. The article provides new model-driven approach to increase reuse of existing solutions in configuration management area. In order to illustrate the approach, there were developed new meta-models that are purposed for development of different configuration management models in context of a model-driven approach. This article provides a simplified example to illustrate models and defines further researches.

1 INTRODUCTION

Software configuration management is a discipline which controls the software evolution process. The main task of process is include only valid and tested items in final version of product (Aiello, 2010), (Berczuk, 2003). In the 2013 report about results of software development projects (Rajkumar, 2013), it was mentioned that only 9% of major projects ended successfully. As one of the main reasons is bad management. Configuration management is one of the disciplines of quality assurance, which is described in the quality standards such as CMMI, ISO 9001 (About CMMI Institute, 2014), (Bamford, 1995) and framework (ITIL Home, 2014). To setup this process, a set of particular tasks should be solved: configuration identification, version control, source code management, build management etc. Usually companies already have tools and scripts to implement mentioned tasks of configuration management. The main challenge is adaptation of these solutions for new projects with minimum additional efforts. In century of improvement of cloud computing and model-driven development, static scripts and solutions for configuration management are not effective (Ragan, 2014). Solutions for software configuration management tasks should be model-driven to increase its reuse. The article introduce a problem with lack of reuse of existing scripts and tools for software configuration management subtasks. The reasons of low reuse are mixtures of different parts of configuration management in one solution or script. For example, company could have script that moves configurations from test instance to production. This script could not be applied in other project with the same set of technologies because it contains hardcodes, source code management, build and installation actions in one place.

The article analyzes trends of configuration management solutions. Based on study of existing solutions, new model-driven approach is developed. Unlike other solutions, new approach is oriented to increase reuse of existing solutions and do not requires to use some special tool. New models provide a way how to develop reuse-oriented solutions for configuration management using well-known tools. Finally, conclusions and further researches are given to improve new model-driven approach.

2 RELATED WORKS

As far back as 1992 there was published an article

Bartusevics A. and Novickis L..
Model-based Approach for Implementation of Software Configuration Management Process.
DOI: 10.5220/0005228701770184
In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2015), pages 177-184
ISBN: 978-989-758-083-3
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

(Dart, 1992) where five future challenges of configuration management are defined. One of the mentioned challenges introduces one of the first attempt to define configuration management by models. In a recent interview with a long-term configuration management specialist (CMCrossroads, 2014) was mentioned the year 1998, when there was an attempt to create a “super tool” to deal with all the tasks at once in configuration management process. In practice this attempt failed, because it was too complicated to use, and programmers were afraid of such a tool of "majesty and mysticism." As the future trend the configuration management specialist (CMCrossroads, 2014) emphasizes the need to enhance trust between configuration management and programmers. Other configuration management experts (Aiello, 2010), (Berczuk, 2003) note that it is necessary to plan the process and only then apply tools for implementation, otherwise solutions will be ineffective and will require to invest a lot of resources inefficiently. Finnaly, (Ragan, 2014) note that solutions should be model-driven and not static to increase its reuse.

Large part of existing researches related to software configuration management uses ideas of model-driven approach (Osis, 2010). The most important task in configuration management is the version control and a significant part of model-driven researches is devoted to version control (Yongchang, 2010), (de Almeida Monte-Mor, 2014), (Toth, 2013). New approaches try to improve version control, in order to better control changes in the product code (Toth, 2013), as well as provide abstract models that can be used in development of new version control systems (Yongchang, 2010), (de Almeida Monte-Mor, 2014). There are also solutions offering an abstract model of configuration management, based on quality standards or specific characteristics of the software development approach (Estublier, 2000), (Ruan, 2003), (Mingzhi, 2008). Usually the proposed approaches are not supported by tools which could allow performing experiments and making sure about benefits.

The following solutions (Pindhofer, 2009), (Calhau, 2012), (Giese, 2009) consider configuration management process as a whole, not just a specific part. Solution, described in (Pindhofer, 2009) provides a definition of configuration management model. The model devoted to support of configuration identification and build management. The solution is focused on projects based on model-driven approach, but there are no recommendations how this approach could be applied in projects with classical development methodologies.

Configuration management principles for the following approach (Giese, 2009) were taken from the ITIL (Information Technology Infrastructure Library) framework and later were created abstract models, from which configuration management process could be created and later the model could be transformed into platform specific model. Although that solution also includes an implementation for proposed model-driven configuration management, it is focused on a single technology (JAVA) and implementation details are not given.

Study (Calhau, 2012) focuses on mutual integration of various configuration management tools. In order to maintain a full configuration management process, it is required a number of tools: version control systems, bug tracking systems, build servers, continuous integration servers and other tools. As practical experience indicates, all tools work separately from each other. The study offers an ontology for configuration management process. This ontology is used as a configuration management model that shows how various configuration management tools should be integrated. The study does not have any specific instructions how the ontology could be applied for particular project. It is not clear what kind of ontology editors are advised to use and how to determine the moment when the changes have to be made.

3 GENERAL APPROACH FOR MODEL-DRIVEN CONFIGURATION MANAGEMENT

The article provides a new model-driven approach for software configuration management. The approach is based on model creation and gradual transformation into the model with a lower level of abstraction. Unlike other solutions, a new approach supports all configuration management tasks and does not impose any specific tool. New approach shows a way how to increase reuse of existing solutions. New meta-models are developed to create models with different levels of abstraction. Additionally, transformation rules are designed to transform one model to other.

New model-driven approach for configuration management is based on assumption that generally configuration management answers the question

"How to transfer certain software changes from one instance into another at the right moment?" Approach assumes explore the literature on configuration management (Aiello, 2010), (Berczuk, 2003), (Upravlenije projektami, 2011), and also take into account personal experience of authors in position of configuration manager. The answers to above mentioned question requires that all changes in software are controlled, and is known a certain moment when the changes are ready to be transferred, there is information about all the instances involved in the process, and is also known the process of transferring the changes. This means that the answer to the question requires solving the basic tasks of configuration management: configuration item identification, version control, configuration account, building and installation management (Aiello, 2010).

There are three models in provided approach:

- *Environment Model (EM)* – provides a model of all environments included in a software development project. A model also contains all flows of software changes between different environments. Environment in context of this model is a infrastructure for particular process of project, for example DEV for development, TEST for testing etc.
- *Platform Independent Action Model (PIAM)* – provides a set of actions needed to apply all flows from Environment Model. The actions are abstract. For example, action "Compile" should be used to compile software from source code, but in this model any details about software technology and compilation algorithm are not known.
- *Platform Specific Action Model (PSAM)* – provides an extended variant of Platform Independent Action Model because actions are fulfilled with details about platform, specific scripts, etc. In current model, action "Compile" already have information provides details of compilation algorithms and platform. In this model all details are known, for example, it could be MAVEN build script for JAVA projects.

General picture of a new model-driven approach provided in Figure 1.

Model-driven approach (Figure 1.) represented as flow with interactions from a software configuration manager. The arrows with numbers mean particular steps of the approach. The first step "1" provides creation of Environment Model from a special meta-model. Software configuration manager builds Environment Model from the set of

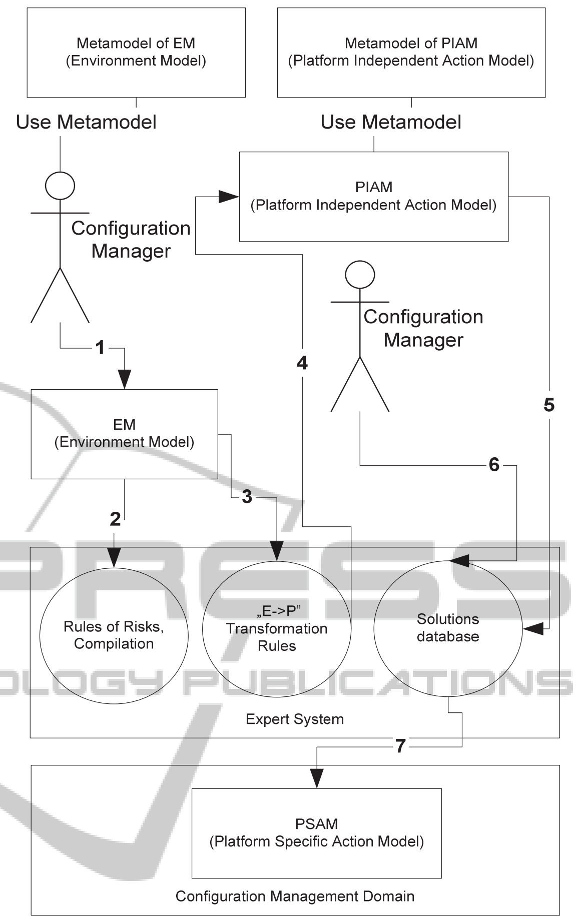
Metamodel of EM (Environment Model)
Metamodel of PIAM (Platform Independent Action Model)
Use Metamodel
Use Metamodel
PIAM (Platform Independent Action Model)
Configuration Manager
Configuration Manager
1
4
5
6
EM (Environment Model)
2
3
Rules of Risks, Compilation
„E->P" Transformation Rules
Solutions database
Expert System
7
PSAM (Platform Specific Action Model)
Configuration Management Domain

Figure 1: Approach of the Model-Driven Software Configuration Management.

components from the mentioned meta-model. During the second step "2", created Environment Model should be compiled by special block in Expert System, named "Rules of Risks, Compilation". Expert System in context of this article is a special warehouse for different blocks of rules. Expert System also contains a database with information about ready solutions of actions. After second step "2", a configuration manager has compiled Environment Model with the description of general configuration management risks. Step "3" explores ready Environment Model by special block of Expert System called "E->P". The main task of "E->P" block is to detect actions needed to apply each flow between environments. During step "4", Platforms Independent Action Model performing from actions defined at step "3" and from meta-model of PIAM. The steps "5" and "6" require the second interaction from configuration manager to analyse ready Platform Independent Action Model and choose solutions for each action from "Solutions

Database". Structure of mentioned database provided in Figure 2.

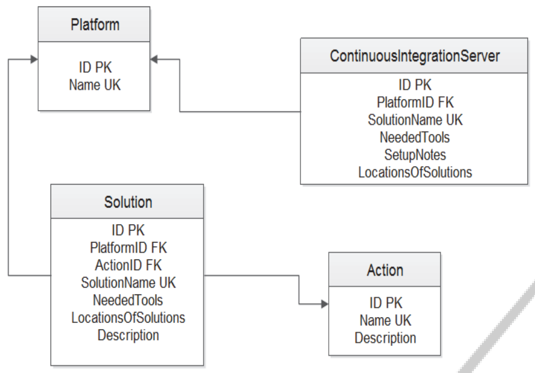

Figure 2: Structure of Solutions Database.

Solutions Database contains all information about all configuration management actions described in PIAM. For example, action "Compile" could have four different solutions to compile executable from source code for the following technologies: Ruby, C++, Oracle, C#. The mandatory requirement is that all solutions are parameterized and do not have dependencies on solutions of other action. For example, compilation script should not know any details about other actions from Platform Independent Model, hardcodes from bug tracking system, names of hosts, absolute paths of storages, etc. All necessary things should be given as parameters. Any solution stored in "Solution Database" has the following attributes:

- *ID* – unique identifier;
- *PlatformID* – reference to platform;
- *ActionID* – reference to action. Table "Action" contains all possible actions and strongly dependent from PIAM meta-model;
- *NeededTools* – set of tools to implement particular solution;
- *LocationsOfSolutions* – information about ready scripts, functions, locations of servers etc.;
- *Description* – notes provide additional information about implementation.

During the last step "7", ready PSAM model should be implemented.

4 META-MODELS FOR EM AND PIAM

4.1 Meta-Model for Environment Model

Table 1 shows all elements of EM meta-model and their attributes. Attributes are filled during simulation by configuration manager and are later used in transformation rules to prepare PIAM model. Meta-model for Environment Model contains the following elements:

- Set of graphic elements – the user works with this set of elements to display graphically all instances and flows of changes between them;
- The algorithm that converts graphic elements into XML format;
- Classes of model elements and their hierarchy. Each element has an abstract class with a list of methods "add". The EM includes logic to ensure proper formation of a environment model. For example, there should not be two identical instances, and the programmer should not make changes in the source code in production environment (PROD);
- Model compilation algorithm. Reads a model structure, creates a object for each element and tries to fill in information with special "add" methods. If the addition of some elements has failed, the algorithm informs the user about compilation error and terminates compilation;
- Set of XML elements. Each graphic element has the appropriate element in XML format with the same name. XML format is necessary, in order the model could be processed by computer.

4.2 Meta-Model for PIAM

The main goal of PIAM is to show all tasks needed to implement all flows of changes in EM. PIAM model has three parts:

ContinuousIntegrationServer - simulates framework, where all configuration management activities take place. The following articles (Aiello, 2010) and (Berczuk, 2003) state that all configuration management activities have to be done from one location and the performance should not depend on a local workstation. The element has the following attributes: "Platform" - a platform where it operates, "Name" -name of the tool, "InstallationNotes" - information about implementation of current framework, "LocationsOfSolutions" – ready sources, such as installation files, scripts, instructions, etc;

Events – copied from Environment Model.

Configuration management actions - actions needed to implement ConfigurationItemFlow. There is a fixed set of actions. Transformation rules "E->P", depending on attributes of ConfigurationItemFlow elements, determine which actions from a fixed set are needed for particular configuration item flow.

The Table 2 shows actions, while in the Table 3

Table 1: EM meta-model elements.

Element name	Description	Attributes
Environment	Instance which contains items, exposed to configuration management.	Name – name.
		Decsription – information about environment.
		CustomerSupportFlag – Instance belonging to the customer.
		DevelopmentFlag – features of developed environment.
		OriginalEnv – feature or environment used in real process or just testing various transfers of changes.
		OriginalEnvName – appropriate original name of the environment, in case the given environment is used only for configuration test purposes.
ConfigurationItemFlow	Configuration item flow.	Name – name.
		Source - environment where configuration comes from.
		Sequence - sequence number.
		Goal - environment where configuration goes to.
		Description – information about current flow.
Event	Simulates transfer of changes between two environments used in project. One Event could have one and more ConfigurationItemFlow.	Name – name.
		ConfigurationItemFlows – flows of changes.
		Description – description.
		AllChangesMoveFlag – sign that either all configuration or only a part of the change is transferred.

can be seen attributes inherent in each activity. Values of attributes in PIAM model should be empty.

Table 2: Actions of PIAM model.

Name	Description
DEVELOPMENT	A programmer makes changes in source code. This process is controlled by configuration management.
COMMIT_CHANGES	Activity, which ensures any changes to version control, and defines the rules for proper saving of changes.
PREPARE_BASELINE	Provides management of source code and different branches.
COMPILE_BUILD	Build from certain baselines in version control system.
INSTALL_BUILD	Deploy ready build at environment.
PRODUCT_DELIVERY	Product delivery preparation and shipping to the customer so that he could get changes (release) to his environments.
ENV_UPDATE_NOTIFIC ATION	It performs necessary registration and administrative actions when the customer has installed the product into environment.

Table 3: Attributes of actions in PIAM model.

Attribute	Description
Platform	Reference to platform.
SolutionName	Unique name.
NeededTools	Tools that are needed for implementation.
LocationsOfSolutions	Sources of existing solutions (scripts, descriptions, installation files etc.)
Description	Description which can help to implement a solution.

4.3 Transformation from EM to PIAM

Compilation algorithm reads XML structure of EM model and establishes class hierarchy, where each element has a separate object. Modelling logic is implemented in "add" methods of EM classes, which is a basis for compilation algorithm. If compilation finishes successfully, the established class structure comes into Expert System block "E->P". This block contains 12 transformation rules. On the left-hand side is stored a specified condition of configuration item flow (ConfigurationItemFlow), while the right-hand side has actions from Table 2. Transformation algorithm reads the information about each Event, each ConfigurationItemFlow and compares this information with all 12 rules. In case of matching, actions from appropriate right-hand side rule are added to a particular ConfigurationItemFlow. The algorithm adds found actions to PIAM model and moves to the next ConfigurationItemFlow. As a result, ready PIAM model is available for configuration manager. Table 4 shows some examples of transformation rules.

Table 4: Examples of "E->P" rules.

Condition of ConfigurationItemFlow attributes (IF)	Actions from PIAMsource, description (THEN)
ownerIsActor() == true	DEVELOPMENT COMMIT_CHANGES Provede control of development and save changes in version control system.
ownerIsActor() == false ownerIsOriginalEnvironment() == false getConfigurationItemFlowSequence() == 1 ownerCustomerSupportFlag() == true	COMPILE_BUILD PRODUCT_DELIVERY ENV_UPDATE_NOTIFICATION If you need to transfer configuration from the testing environment to the environment which is maintained by the customer, and it is the first flow within a particular Event, it is necessary to compile the product, arrange the delivery of compiled product with all the concurrent documentation and await confirmation that the customer has successfully installed this product.

5 EXAMPLE OF MODELS IN MODEL-DRIVEN APPROACH FOR CONFIGURATION MANAGEMENT

Figure 3 shows EM, PIAM and PSAM models in the case when developed changes have to be transferred from DEV environment to TEST. Another additional instance Pre_TEST will be introduced, where each new release is tested. This will reduce the risk of unavailability of TEST instance. Thus, EM model has three instances: DEV, Pre_TEST and TEST, one Event that provides transfer of changes from DEV to TEST. The Event has two ConfigurationItemFlows. During the first flow configuration is transferred to Pre_TEST instance, and if the transfer is successful, then in the second flow the same configuration is transferred to the original test environment. Then "E->P" transformation rules detect configuration management actions, which ensure all the above-mentioned flows. In the first flow happens the merging of corresponding new changes in TEST environment source code (PREPARE_BASELINE), then from obtained code is created product build (COMPILE_BUILD), and finally deployed to Pre_TEST instance (INSTALL_BUILD). If the first flow has been successful, the same build should be applied for TEST instance (INSTALL_BUILD). Next, solutions from Solutions Database are chosen for the following tasks: PREPARE_BASELINE, COMPILE_BUILD and INSTALL_BUILD. By fulfilling the attributes is obtained PSAM model. The model specifies which system of version control should be used for version control, which branches

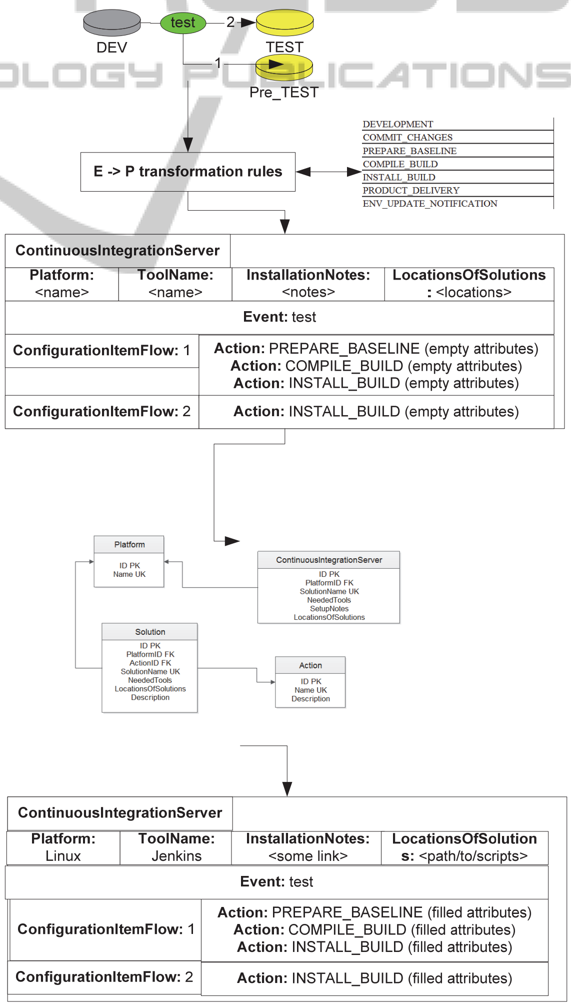
DEV
test
2
1
TEST
Pre_TEST
E -> P transformation rules
DEVELOPMENT
COMMIT_CHANGES
PREPARE_BASELINE
COMPILE_BUILD
INSTALL_BUILD
PRODUCT_DELIVERY
ENV_UPDATE_NOTIFICATION
ContinuousIntegrationServer
Platform: <name>
ToolName: <name>
InstallationNotes: <notes>
LocationsOfSolutions : <locations>
Event: test
ConfigurationItemFlow: 1
Action: PREPARE_BASELINE (empty attributes)
Action: COMPILE_BUILD (empty attributes)
Action: INSTALL_BUILD (empty attributes)
ConfigurationItemFlow: 2
Action: INSTALL_BUILD (empty attributes)
Platform
ID PK
Name UK
ContinuousIntegrationServer
ID PK
PlatformID FK
SolutionName UK
NeededTools
SetupNotes
LocationsOfSolutions
Solution
ID PK
PlatformID FK
ActionID FK
SolutionName UK
NeededTools
LocationsOfSolutions
Description
Action
ID PK
Name UK
Description
ContinuousIntegrationServer
Platform: Linux
ToolName: Jenkins
InstallationNotes: <some link>
LocationsOfSolutions: <path/to/scripts>
Event: test
ConfigurationItemFlow: 1
Action: PREPARE_BASELINE (filled attributes)
Action: COMPILE_BUILD (filled attributes)
Action: INSTALL_BUILD (filled attributes)
ConfigurationItemFlow: 2
Action: INSTALL_BUILD (filled attributes)

Figure 3: Visual example of models of software configuration management.

should be established, with what kind of scripts or tools should be implemented for merge and etc.

6 CONCLUSIONS AND FURTHER RESEARCHES

The article offers a new, model-driven approach for implementation of configuration management process. Unlike other approaches, new one is focused on gradual implementation of the process using existing solutions repeatedly. In order to implement new approach, there have been developed meta-models for creating EM and PIAM models, as well as transformation rules that provide creating PIAM model from EM. In addition database structure was designed for management solutions of configuration management actions. All of these innovations were illustrated by a simplified test example.

New model-driven approach provides saving of resources for implementation and maintenance, improving the re-use option of existing solutions, and also provides a link between the formal process (Environment Model) and technical implementation (PSAM in Configuration Management Domain). This approach will also improve solution maintenance for configuration management, because all of the solutions will be stored centralized and according to common principles.

The further research with the help of experiments should determine the benefits of new approach in various software development projects. Firstly should be established criteria which help measure the indicators of configuration management before and after implementation of a model-driven approach. With the help of the experiment it will be extremely important to determine how many resources is required for refactoring the existing solutions to parameterized condition, because it will give an idea whether some upgrades in the models are needed, or a new meta-model, such as Continuous Integration Server Framework or some other.

The offered model-driven approach is abstract, because it provides only principles of operation and the essence of the model. However, EM, PIAM, and PSAM model implementation in theory may be different from that, what was offered by the authors of this paper. Hence, the article is concluded with the hope that the new approach will contribute to the emergence of a new idea with regard to configuration management process modelling and reuse of solutions.

ACKNOWLEDGEMENTS

The research has been partly supported by the project eINTERASIA “ICT Transfer Concept for Adaptation, Dissemination and Local Exploitation of European Research Results in Central Asia's Countries”, grant agreement No. 600680 of Seventh Framework Program Theme ICT-9.10.3: International Partnership Building and Support to Dialogues for Specific International Cooperation Actions – CP-SICA-INFSO.

REFERENCES

About CMMI Institute. 2014. (ONLINE) Available at: http://whatis.cmmiinstitute.com/about-cmmi-institute. (Accessed 02 September 2014).

Bamford, R., 1995. *Configuration Management and ISO 9001*. Software Systems Quality Consulting, DO-25 V6, 7.

Calhau, R., (2012). *A Configuration Management Task Ontology for Semantic Integration*. In 27th Annual ACM Symposium on Applied Computing. Italy, March 26 - 30, 2012. IEEE Digital Library: ACM. 348-353.

CMCrossroads. 2014. *How Configuration Management Is Changing: An Interview with Joe Townsend*. (ONLINE) Available at: http://www.cmcrossroads.com/interview/how-configuration-management-changing-interview-joe-townsend?page=0%2C0. (Accessed 02 September 2014).

Dart, S., 1992. *The Past, Present, and Future of Configuration Management*. CMU/SEI-92-TR-8, 1, 25.

de Almeida Monte-Mor, J., (2014). *GALO: A Semantic Method for Software Configuration Management*. In Information Technology: New Generations (ITNG), 2014. USA, 7-9 April, 2014. ITNG: IOT360. 33 - 39.

Delaet, T., (2007). *PoDIM: A Language for High-Level Configuration Management*. In Large Installation System Administration Conference, LISA 2007. USA, November 11-16, 2007. IEEE Digital Library. 01/2007.

Estublier, J., (2000). *Software configuration management: a roadmap*. In ICSE '00 Conference on The Future of Software Engineering. USA, June 4-11, 2000. IEEE Digital Library: ACM. 279 - 289.

Giese, H., (2009). *A Model-Driven Configuration Management System for Advanced IT Service Management*. In International Conference on Model Driven Engineering Languages and Systems (MoDELS 2009). USA, October 4 - 9, 2009. IEEE Digital Library: 4th International Workshop on

Models. 300-310.
ITIL Home. 2014. (ONLINE) Available at: http://www.itil-officialsite.com. (Accessed 02 September 2014).
Janis Osis, 2010. *Model-Driven Domain Analysis and Software Development: Architectures and Functions (Premier Reference Source)*. 1 Edition. IGI Global.
Mingzhi, M., (2008). *A New Component-Based Configuration Management 3C Model and its Realization*. In Information Science and Engineering, 2008. China, December 20-22. 2008. IEEE Digital Library: IEEE. 258 - 262.
Pindhofer, W., 2009. *Model Driven Configuration Management*. M.Sc.. Wien: Wien University.
Q&A and Advice. 2014. CMCrossroads. [ONLINE] Available at: http://www.cmcrossroads.com/. [Accessed 02 September 2014].
Rajkumar, G., 2013. *The most common factors for the failure of software development project*. The International Journal of Computer Science & Applications (TIJCSA), 1, 11.
Robert Aiello, 2010. *Configuration Management Best Practices: Practical Methods that Work in the Real World. 1 Edition*. Addison-Wesley Professional.
Ruan, Li., (2003). *A new configuration management model for software based on distributed components and layered architecture*. In Parallel and Distributed Computing, Applications and Technologies, 2003. China, August 27-29, 2003. IEEE Digital Library: IEEE. 665 - 669.
Smashing Magazine. 2014. *7 Version Control Systems Reviewed*. (ONLINE) Available at: http://www. smashingmagazine.com/2008/09/18/the-top-7-open-source-version-control-systems/. (Accessed 02 September 2014).
Stephen P. Berczuk, 2003. *Software Configuration Management Patterns: Effective Teamwork, Practical Integration. Edition*. Addison-Wesley Professional.
Toth, Z., (2013). *Using Version Control History to Follow the Changes of Source Code Elements*. In Software Maintenance and Reengineering (CSMR), 2013. Italy, March 5–8, 2013. IEEE Digital Library. 319 - 322.
Upravlenije projektami OpenQuality.ru. 2011. (ONLINE) Available at: http://experience.openquality.ru/ software-configuration-management/. (Accessed 02 September 2014).
Yongchang, R., (2010). *Fuzzy Decision Analysis of the Software Configuration Management Tools Selection*. In ISCA 2010. France, 19-23 June, 2010. Information Science and Engineering (ISISE): ACM. 295 - 297.
Zeroturnaround.com. 2014. *DevProd Report Revisited: Continuous Integration Servers in 2013*. (ONLINE) Available at: http://zeroturnaround.com/rebellabs/ devprod-report-revisited-continuous-integration-servers-in-2013. (Accessed 02 September 2014).
Ragan Tracy, (2014). *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.