Semantic Web Technologies and Model-Driven Approach for the Development and Configuration Management of Intelligent Web-Based Systems

Arturs Bartusevics, Andrejs Lesovskis, Leonids Novickis

Abstract- The study provides the model-driven approach for implementation of software configuration management. Provided approach uses technologies of Semantic Web to improve transformations between different models. Software configuration management is a discipline that controls software evolution process and helps to make valid builds of intelligent web-based systems. Nowadays, high level of agility requires to setup process of software configuration management as soon as possible. Provided approach helps to decrease process implementation time by reuse of existing source code for new processes.

Keywords— Software Configuration Management, Model-Driven Approach, Semantic Web.

I. Introduction

Software Configuration Management (SCM) is the discipline that controls the evolution of large and complex software systems. SCM tools and systems vary and range from small tools such as RCS (Revision Control System) over medium-sized systems such as Subversion to large-scale industrial systems such as Adele ClearCase.

Nowadays software configuration management is not only challenge to choose optimal system for source code management. Complex software development projects with multiple mutually dependent components and high level of agility require two important things: firstly, in context of software configuration management, many tasks should be implemented such as source code manage managements, version control, build and deploy management, accounting of statuses of items, etc.; secondly, the mentioned implementation should be ready as soon as possible because agile methodologies require frequent releases of new versions of product.

Some of the most common problems in the area of software configuration management are the following:

- Use of multiple different configuration management tasks in a single solution. For example, use of a script that performs source code management, build management, and installation management tasks. Such multifunctionality makes this script specific for one particular project and makes it impossible to reuse it without some modifications.
- Lack of approaches and recommendations on how to design reuse-oriented solutions for configuration management that could be used in the other projects without additional customizations to save up time and resources.

Reusable configuration management solutions should be parameterized and structured by different tasks. It means that, for example, solutions on how to build the product from the source code should be independent from the other tasks like source code management or installation management. The mentioned product build solution should receive a set of parameters and return an executable or an error message. It should not contain any details or hardcoded information like the location of the source code or the address of the server where the executable should be installed.

The paper describes a new model-based approach for implementation of software configuration management. Unlike the other approaches, it is not oriented to any particular tool or script that “should solve any problem” but describes the steps how to increase the reuse of the existing solutions. This approach is independent from the tools being used for tasks like source code management, continuous integration, bug tracking, build management, etc. as it only defines a way to make a solution reusable.

Authors also investigate how the Semantic Web technologies like OWL and SPARQL could be used to improve this approach and to perform transformations between different levels of models.

The paper is structured as follows: the second section describes the work done by other researchers in the fields of Software Configuration Management and Semantic Web technologies. The third section covers the use of the Semantic Web technologies (like RDF, OWL, etc.) in the model-driven Software Configuration Management and potential advantages they could bring. The fourth section contains the detailed description of the proposed model-driven EAF methodology. Fifth section is concerned with how the Semantic Web

technologies could be used to improve EAF methodology. The last section is related to the results of the experimental assessment of the EAF methodology in 5 different software projects.

II. Related Works

Long term expert in software build management Tracy Ragan in her paper [1] writes that solutions for software configuration management should be model-driven, not static script-oriented as it was normal in 20th century. That paper outlined some of the reasons why model-driven approaches could be better for modern software configuration management processes.

Firstly, a lot of software products are supported by cloud computing technologies where the things like server names, absolute paths, and other information required for static scripts and specific platforms are unknown. In this case, model-driven approach could provide a way from planning to implementation of software configuration management in virtual environments [1]. There are many tools can configure builds and deployments for complex products within a few hours without writing huge and complex static scripts for particular platforms [2].

Secondly, model-driven approach helps to reduce the human factor during selecting solutions for particular task of software configuration management. Usually traditional implementation of software configuration management contains two major steps: process planning and development of static scripts for the planned process. There are risks related to writing source code for scripts, because sometimes executable source code is not in consistence with initial planning. Model-driven approach could reduce these risks by generating source code for configuration management automatically [1][2][4].

Unfortunately, modern tools mentioned in [2] are oriented only for one task of software configuration management: build and deployment management. But it is not possible to prepare a valid build with any tool without correct source code management [3][4][5]. Additionally, tools that mentioned in paper [2] help to improve builds and deployments, but can not fix successful use cases to reuse them in new projects. These tools require the acquisition of additional knowledge and financial investments. However, sometimes enterprises already have tools for software configuration management that they trust and rely on. Because of that these companies are looking for approaches and methodologies that could increase reuse of the existing solutions.

There are some novel approaches related to implementation of software configuration management using models. The study [3] provides semantic integration of different tools. Ontologies are used to create concept of each tool using in software configuration management process.

The paper [4] presents software configuration management model based on ITIL framework. The model is theoretical and no any suggestions about increasing reuse of existing solutions are given. The main advantage of approach [4] is based on well-known framework that works in real world, but lack of suggestions about tools and approaches that could be used for implementation makes provided approach perspective in theory but not trusted in practice.

Approach described in work [5] provides a method for selecting optimal software configuration management tool by using artificial intelligence methods. The method seems very useful from the theoretical point of view. Empirical evidence provided in [5] shows that method could really improve source code management and version control in particular software development project. As a main disadvantage is a fact, that theory of fuzzy logic is relative difficult for understanding.

In context of software configuration management, very important task is a management of source code and version control. The study [6] describes the approach related to improvement of version control. Nowadays, majority of version control tools supports management of source code. However, there is a lack of approaches that provide version control for model-driven software development. Approach presented in work [6] describes a model of universal version control system that could be used for code and model management.

The current paper provides approach for implementation of software configuration management process. Provided approach is an improved version of the researches described in works [7][8]. Practical assessment of the first version of the approach outlined some disadvantages. In the improved version of approach described in current paper, all useful ideas will be taken from studies [3][4][5][6] and conclusions from the practical experiments of approaches [7][8].

Unlike other approaches, methodology described in this paper:

- Does not impose to use any specific tools for software configuration management process,
- Increase reuse of existing solutions for software configuration management,
- Defines full cycle of models related to step-by-step implementation of software configuration management using existing solutions,
- Transformations between models are improved with semantic web technologies.
- Has an abstract views that allows to design new implementation of provided approach.

This study is not the first attempt to introduce the Semantic Web technologies into software configuration management. In a related study, Falbo [13] proposed an SCM ontology that was used to establish a common conceptualization about the SCM domain in order to support SCM tools integration.

III. Semantic Web technologies and model-driven software configuration management

The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [10]. It is based on the idea of having data on the Web defined and

linked in a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various applications.

The Semantic Web is composed of a set of technologies, and it can be defined as a symbiosis of Web technologies and knowledge representation. Semantic Web technologies can be used in a variety of application areas; for example: in data integration, whereby data in various locations and various formats can be integrated in one, seamless application; in cataloging for describing the content and content relationships available at a particular Web site, page, or digital library; by intelligent software agents to facilitate knowledge sharing and exchange; in content rating; in describing collections of pages that represent a single logical "document"; for describing intellectual property rights of Web pages, and in many others [11].

Authors think that the Semantic Web technologies can be efficiently utilized in the software configuration management to ease and improve efficiency of processes like data integration and reuse, transformation, and searching.

Ontologies serve as a key enabling technology for the semantic software configuration management. Ontologies are developed to provide a machine-processable semantics of information sources that can be communicated between different agents (software and humans). Ontology is an explicit formal specification of a shared conceptualization. 'Conceptualization' refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon. 'Explicit' means that the type of concepts used and the constraints on their use are explicitly defined. 'Formal' refers to the fact that the ontology should be machine readable. Hereby different degrees of formality are possible [12].

The type of knowledge used to describe SCM field is very hard to represent formally using traditional approaches. Organizations use different proprietary solutions that make it much harder to reuse encoded software configuration information across different software projects. The lack of a standard for compatible encoding of configuration data is one of the major SCM problems. Web Ontology Language (OWL), a family of knowledge representation languages for ontology authoring and one of the key Semantic Web building blocks, provides developers with features and opportunities that can be used to solve this problem.

OWL is an ontology language designed for use in the Semantic Web and is the language recommended by the W3C for this use. OWL DL and OWL Lite semantics are based on Description Logic (DL). OWL 2 exhibits the desirable features of Description Logics, including useful expressive power, formal syntax and semantics, decidability, and practical reasoning systems, resulting in OWL 2 providing effective ontology representation facilities.

Ontologies provide software developers with a standard and powerful way of representing knowledge not only the configuration management tasks but about software engineering project in general.

Besides the general benefits of formal specification that can be gained by encoding the software configuration knowledge, the main advantage of using OWL ontologies is an ability to provide a fully automated procedure to detect inconsistencies in the configuration via standard reasoning engines (for example, Pellet or RacerPro). The reasoning service can not only determine, if a certain configuration is valid or not, but it also provides justifications for such decision (i.e. it lists the reasoning steps that lead to this decision).

One of the key benefits of the use of the Semantic Web technologies is that they provide means to reason and query over semantically annotated metadata from the software configuration models. Reasoning provides an opportunity to perform an inference.

Inference is a process to infer a new relationship from the existing resources and some addition information in form of set of rules. Inference base technique is also used to check data inconsistency at time of data integration. The inference engine can be described as a form of finite state machine with a cycle consisting of three action states: match rules, select rules, and execute rules [12]. The use of DL reasoners allows OWL ontology applications to answer complex queries and to provide guarantees about the correctness of the result. This is obviously of crucial importance when ontologies are used in safety critical applications.

Some of the features that can be provided by a standard Description Logic reasoner are the following [9]:

- Consistency checking – ensures that an ontology does not contain any contradictory facts;
- Concept satisfiability - determines whether it is possible for a class to have any instances;
- Classification - computes the subclass relations between every named class to create the complete class hierarchy;
- Realization - computes the direct types for each of the individuals.

The reasoning is especially important when developers are dealing with different versions of software application to support various machines and/or operating systems and in the scenarios where the constantly changing requirements are different for different target groups. For example, it is possible to detect that individual doesn't belong to a certain class or doesn't satisfy constraints and/or restrictions of available configurations.

Another important aspect of using OWL ontologies is that every document or resource can be uniquely identified and referenced using Universal Resource Identifier (URI). This makes tasks like configuration management planning much easier

IV. EAF Methodology for Implementation of Software Configuration Management Process

Methodology provided in this section is an improved version of works [7][8] and is related to decreasing

implementation time of software configuration management by reuse of existing solutions. The main principles of novel methodology are the following:

- Ready implementation of software configuration management process is a source code that could be executable only from particular software configuration management server. It could be one of well-known continuous integration servers like Bamboo, Jenkins, CruiseControl etc.
- The main result of EAF methodology is generated source code for software configuration management process.
- Generation of the source code for software configuration management is automated process that uses models and existing executable units of source code.

The name of provided methodology (Environment -> Action -> Framework) provides the main steps for generating source code for software configuration management:

- *Defines all environments in particular software development project.* Environments in context of EAF methodologies is a set of servers, applications, virtual machines needed to use software. For example, web application needs database server and application server to make this application ready to use. Usually, the scope of particular environment is particular process in software development project. For example, DEV environment for development, TEST for testing etc. During the first step of EAF methodology, all environment in project should be defined and all transfers of changes between these environments. The mentioned step is formalized by Environment Model, described in [7].
- *Defines all actions needed to apply Environment Model from the previous step.* In context of this work, actions are tasks of general software configuration management process. For example, to move software changes from DEV environment to TEST environment, the following actions could be defined:
  - PREPARE_BASELINE: merge changes of source code from development to test branch;
  - BUILD: run builds scripts or tools to make executable from prepared source code;
  - INSTALL: deploy ready executables to applications servers.

  In the previous versions of provided methodology [7][8], there are two models: Environment Model and Platform independent Action Model. Improved EAF methodology contains merged variant of mentioned models, called PIEM (Platform Independent Environment Model). This model also contains all environments and flows of changes between them, but additionally, configuration manager could define actions needed to apply transfers of changes. Example of PIEM model is given in Fig.1.

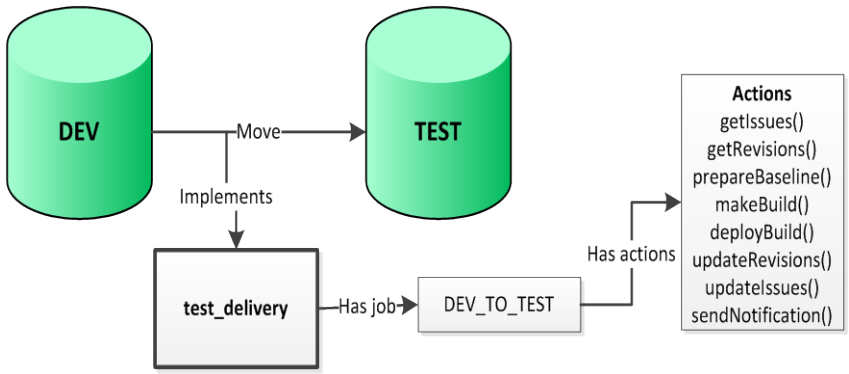

Fig. 1 Example of PIEM model

- Choose the Framework for each action defined in PIEM model. The framework in context of this paper is a set of executable units of source code for implementation of particular action form PIEM model. During this step, configuration manager chooses implementation for each PIEM action from Solutions Database. Solution Database is a structure there all solutions for configuration management in particular enterprise are stored. The structure of Solution Database is improved version that have been provided in [7][8]. The structure of Solution Database provided in Fig. 2.

ISBN: 978-1-61804-285-9

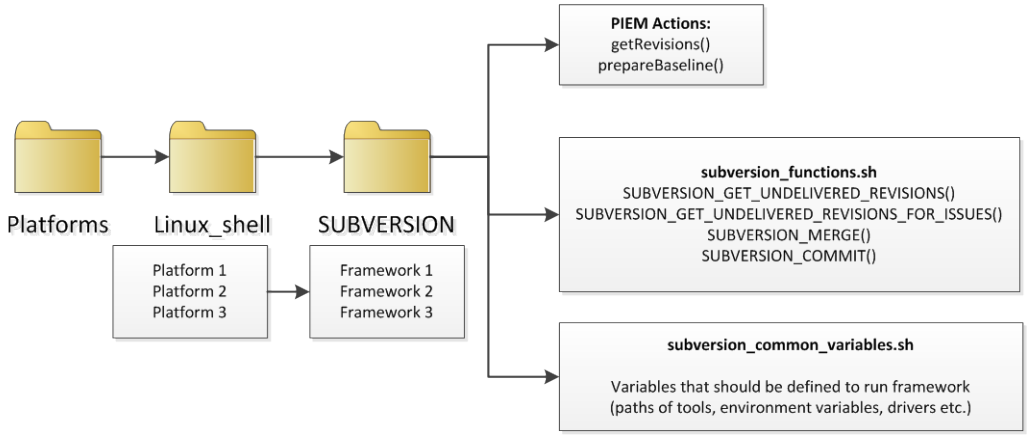

Fig. 2 Solution Database

Solution database provided in Fig. 2 illustrates principles of framework with Subversion example. PIEM model shows actions of configuration management but after actions are defined, configuration manager have to choose a framework for each action. For example, for action "prepareBaseline()" Subversion framework could be selected. Configuration manager should choose platform for implementation and framework. After framework for particular action is defined, Solution Database provides notes about variables that should be defined to activate framework. Each framework has a set of functions that could be called from other scripts and tools. So, during implementation of software configuration management in particular project, only project specific parts should be developed. Significant part of the source code could be taken from framework. Fig. 3 contains an overview of all steps and principles of provided EAF methodology.

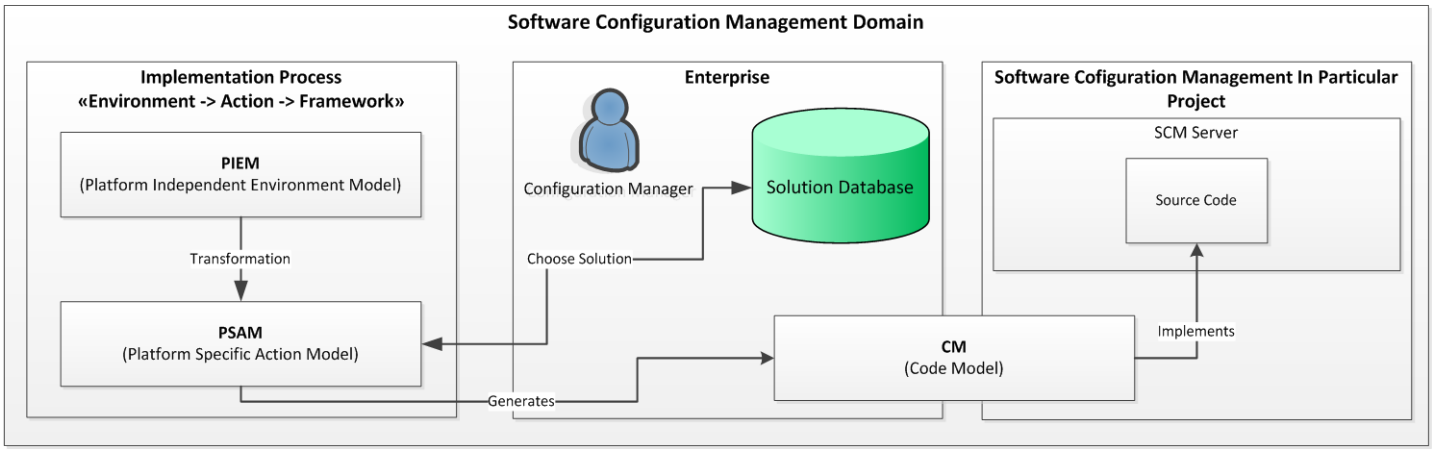

Fig. 3 EAF methodology in enterprise

Fig. 3 contains all main steps of EAF methodology. During the implementation process configuration manager makes environment model (PIEM), defines actions of software configuration management. Ready PIEM model should be fulfilled with information about frameworks. Configuration manager chooses frameworks from Solution Database for each actions. Platform Specific Action Model contains implementation details. This model could be transformed to Code Model and last one could be implemented in software configuration management server to support software configuration management process for particular project.

V. Applying Semantic Web Technologies to the EAF Methodology

Authors think that EAF methodology could greatly benefit from the use of the Semantic Web technologies not only in terms of improved efficiency but could also potentially provide additional functionality.

In their research [14], Arantes et al came to the conclusion that ontology from [13] lacked some important concepts mostly related to change and version control. Therefore, they presented what they called an evolution of this ontology that introduced a few new concepts and a taxonomy of change control actions. The new version features concepts like Repository, Branch, Version, Artifact, Change, etc.

Authors believe that Arantes et al' SCM ontology could be used in the proposed EAF methodology as a base ontology. That would allow the reuse of the valuable expert knowledge encapsulated within this ontology. However, it is necessary to correspondingly modify it according to the needs of the EAF methodology. One of the main reasons for these changes is that the ontology was not designed with a good reasoning support.

ISBN: 978-1-61804-285-9

The Semantic Web Rule Language is a language for the Semantic Web that can be used to express rules as well as logic, combining OWL DL or OWL Lite with a subset of the Rule Markup Language [15]. SWRL complements DL by providing the ability to infer additional information in DL ontologies, but at the expense of decidability. SWRL rules are Horn clause-like rules written in terms of DL concepts, properties, and individuals. SWRL includes a high-level abstract syntax for Horn-like rules in both the OWL DL and OWL Lite sublanguages of OWL.

An SWRL rule is composed of an antecedent (body) part and a consequent (head) part, both of which consist of positive conjunctions of atoms. The SWRL rule syntax is the following:

antecedent ⇒ consequent

where both antecedent and consequent are conjunctions of atoms written a1 ∧ ... ∧ an. For example, we can say that if a is a parent of b and b is a parent of c, then a is also is a parent of c using the following rule:

parent(?a, ?b) ∧ parent(?b, ?c) ⇒ parent(?a, ?c)

SWRL atom can be either a class, an object property, a data type, a data type property, or a built-in. A rule is satisfied by an interpretation if every binding that satisfies the antecedent also satisfies the consequent.

SWRL has already been successfully used in the quite related field of Network Access Control Configuration Management [16]. This experience suggests that SWRL rules can also be used to implement ontology-based SCM model transformation rules (for example, transformation of PSAM to Code Model).

Given the rich SCM ontology, SWRL provides developers with an opportunity to define the resilient rules for different kinds of model transformation scenarios all while inferring possible new knowledge.

VI. Practical Assessment of the EAF Methodology

For the practical assessment, software configuration management process had been implemented at 5 different software development projects by provided EAF methodology. The projects are the following:

- Project 1: Maintenance of Oracle E-Business Suite system. Development technologies are Oracle Forms, Oracle ADF, PL/SQL, JAVA. Version control system is Subversion, bug tracking system is JIRA and continuous integration server is Bamboo.
- Project 2: Implementation of Oracle Customer Care and Billing Utilities system. Development technologies are Oracle CC&B, PL/SQL, Java, Cobol. In this project also Subversion and JIRA are used, but continuous integration server is Jenkins.
- Project 3: Development of Web-Based ERP (Enterprise Resource Planning) system based on Ruby On Rails platform. In this project Git system have been used for version control. For bug tracking and continuous integration also JIRA and Jenkins have been used.
- Project 4: Development of custom billing system based on Oracle Application Development Framework. This project have been used the same set of tools as Project 1.
- Project 5: Development of Web-services for ERP system using Oracle SOA Suite 11g platform. This project also have been used Subversion as version control system, JIRA for bug tracking and Jenkins for continuous integration.

To underline benefits of EAF methodology, implementation time of software configuration management is fixed and compared by implementation time by old methods (without EAF). The difference between old and new implementation time provided at percent. To save up daily processes of mentioned projects, experimental implementation of configuration management by EAF are completed at new (parallel) software configuration management servers. The Fig. 4 provides overview of difference between implementation time of software configuration management by old methods and by EAF methodology. Difference is provided in percent, for example value '-10%' mean that implementation time by EAF methodology of later for 10%, but value '+15%' means that implementation by EAF methodology is not useful because it needs for 15% more time.

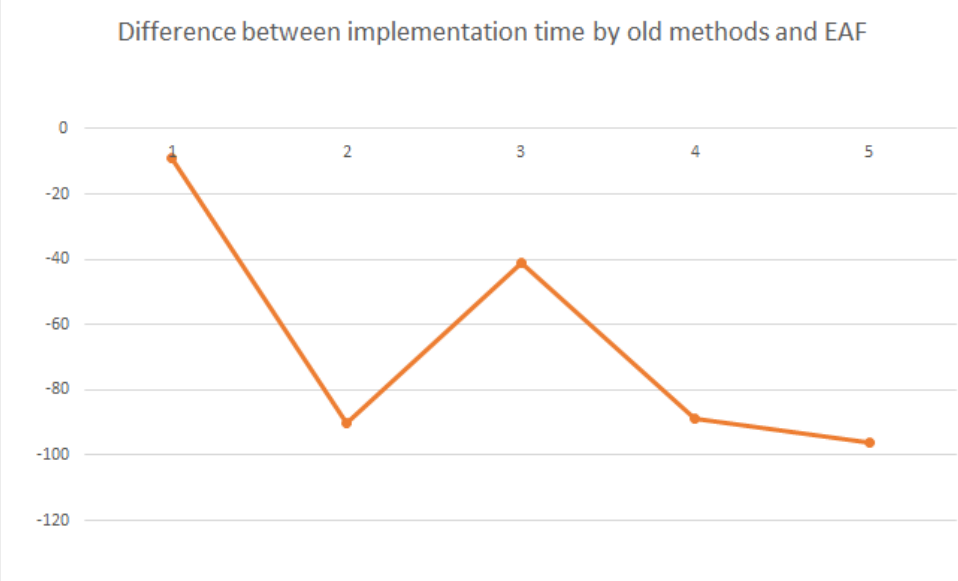

Fig. 4 Difference of implementation time by old methods and EAF

Fig. 4 provides that the first experiment at “Project 1”, while Solution database is empty, benefits of EAF methodology in context of implementation time is only 9%. However, implementation of software configuration management by EAF at Project 2 is for 90% later because Solution Database already has a set of reusable solutions for similar set of tools (Subversion, JIRA). Implementation time at Project 3 is only for 41% later, because Solution Database does not contain framework for Git version control system. The benefits at projects 4 un 5 are great (89% and 96%), because at this moment Solution Database contains all reusable Frameworks for Subversion, Git, JIRA, Jenkins, Hudson. During implementation of software configuration management, mostly existing functions from frameworks are used. In this case, only a small part of source code for particular software configuration management process should be developed.

The main trend provided at Fig. 4 shows that benefits from EAF is relatively small while Solution Database is empty, but if mentioned database contains framework for all most used tools at particular enterprise, implementation time of software configuration management could be for 80% - 95% later.

VII. Conclusions

The study provides new model-driven approach for implementation of software configuration management. The main scope is to increase reuse of existing solutions and reduce efforts to implement the process in other projects. General picture and principles of new EAF approach are provided, Platform Independent Environment Model and Solution Database with example are introduced. Finally, experiments of implementation of software configuration management by EAF methodology are provided.

Results of this work were used in Latvian Council of Science project "Approach and Generic Methodology for Development of Applied Intelligent Software Based on Artificial Intelligence, Modelling, and Web Technologies" (project leader prof. J. Grundspenkis) and in European Commission 7th Framework project "eINTERASIA "ICT Transfer Concept for Adaptation, Dissemination and Local Exploitation of European Research Results in Central Asia's Countries"" (project coordinator prof. L. Novickis).

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

- Develop additional criteria that evaluate models benefits in software development projects not only from point of implementation time,
- Based on developed criteria, evaluate benefits of designed models,
- Develop criteria to assess whether the developed model-driven approach for configuration management implementation corresponds to guidelines of ISO/IEC 15504, ITIL, CMMI standards.
- Design Code Models and transformation algorithms for other platforms.
- Add and improve tools and frameworks in existing platforms.

The approach provided in this article is abstract and only general stages, kinds of models and basic elements are defined. The authors hope that the new approach will generate new ideas because many useful lessons could be learned from different implementations of this model-driven approach.

Acknowledgment

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

[1] 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.

[2] 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, 2014.

[3] 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.

[4] Giese H., Seibel A., Vogel T., 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.

[5] Yongchang, R., 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., 2010.

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

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

[8] Novickis, L., Bartusevičs, A., Lesovskis, A. 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.

[9] Clark & Parsia, LLC, Pellet Features. Available at: http://clarkparsia.com/pellet/features, 2015.

[10] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). "The Semantic Web", Scientific American, May 2001, p. 29-37. Available at : http://www-sop.inria.fr/acacia/cours/essi2006/Scientific%20American_%20Feature%20Article_%20The%20Semantic%20Web_%20May%202001.pdf

[11] W3C. Semantic Web Frequently Asked Questions. Available from http://www.w3.org/2001/sw/SW-FAQ#swgoals, 2009.

[12] Fensel D. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer, 2003, 162 p.

[13] Falbo, R., A., Calhau, R. F. A Configuration Management Task Ontology for Semantic Integration. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing.ACM, New Yorok, 2012. Pages 348-353.

[14] Arantes, L., D., Falbo, R. D., Guizzardi G. Evolving a Software Configuration Management Ontology. [ONLINE] Available at: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C71AC33F802C1644AB292AFD9268ED9F?doi=10.1.1.95.9969&rep=rep1&type=pdf

[15] Horrocks, I. et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [ONLINE] Available at: http://www.w3.org/Submission/SWRL/

[16] Fitzgerald, William M. and Foley, S. N. and Ó Foghlu, M. (2009) Network Access Control Configuration Management using Semantic Web Techniques. Journal of Research and Practice in Information Technology, 41 (2). pp. 99-117.

Arturs Bartusevics currently is a Doctoral Student at Riga Technical University, the Faculty of Computer Science and Information Technology, the Institute of Applied Computer Systems. He obtained BSc (2008) and MSc (2011) degrees in Computer Science and Information Technology, respectively, from Riga Technical University. His research areas are software configuration management, release building and management process and its optimization. He works at Ltd. Tieto Latvia as a Software Configuration Manager.
E-mail: arturik16@inbox.lv

Andrejs Lesovskis is a Doctoral Student at Riga Technical University, the Faculty of Computer Science and Information Technology. He obtained MSc degree in Computer Science and Information Technology at Riga Technical University in 2009. His research areas are e-Learning and Semantic Web. He works as a researcher at Riga Technical University.
E-mail: andreyl@inbox.lv

Leonids Novickis is a Head of the Division of Software Engineering at Riga Technical University. He obtained Dr.sc.ing. degree in 1980 and Dr.habil.sc.ing. degree in 1990 from the Latvian Academy of Sciences. He is the author of 180 publications. Since 1994, he is regularly involved in different EU-funded projects: AMCAI (INCO COPERNICUS, 1995-1997) – WP leader; DAMAC-HP (INCO2, 1998-2000), BALTPORTS-IT (FP5, 2001-2003), eLOGMAR-M (FP6, 2004-2006) – scientific coordinator; IST4Balt (FP6, 2004-2007), UNITE (FP6, 2006-2008) and BONITA (INTERREG, 2008-2012) – RTU coordinator; LOGIS, LOGIS-Mobile and SocSimNet (Leonardo da Vinci) – partner, eINTERASIA (FP7, 2013-2015)- project coordinator. He was an independent expert of IST and Research for SMEs in FP6 and FP7. He is a corresponding member of the Latvian Academy of Sciences and an elected expert of the Latvian Council of Science. His research fields include Web-based applied software system development, business process modeling, e-learning and e-logistics.
E-mail: lnovickis@gmail.com