Model-Driven Software Configuration Management and Semantic Web in Applied Software Development

ARTURS BARTUSEVICS, ANDREJS LESOVSKIS, LEONIDS NOVICKIS
Faculty of Computer Science and Information Technology
RTU
Kalku Street 1, Riga LV-1658, Latvia
LATVIA
arturik16@inbox.lv, andreyl@inbox.lv, lnovickis@gmail.com

Abstract: - Nowadays one of the most important factors of software development is the enhancement of agility. The reason behind this is that modern software development is focused on delivering software to customers as quickly as possible. Software configuration management (SCM) is a set of practices, tools, and techniques designed to control software evolution. In most cases, companies already have the tools and solutions needed to achieve the mentioned goals and the main challenge is to find out how to adopt and implement these tools and solutions in the new projects. This paper presents a new model-driven approach to perform software configuration management using a set of models. Unlike other solutions, this approach is designed to increase the reuse of the existing solutions without any restrictions on the tools being used. Models that represent software configuration management from different levels of abstraction are transformed from one level to another. Authors also investigate how the use of Semantic Web technologies could improve this approach and what benefits they could provide. Lastly, the approach is evaluated in terms of applicability in the field of applied software and further works are defined.

Key-Words: - Software Configuration Management, Model-Driven Approach, Semantic Web

1 Introduction

Nowadays software configuration management is challenged not only with problems like choice of a version control system for particular project and definition of an optimal branching strategy. Analyses of the most popular trends in software development area allow to define the requirements for fast and high quality releases [1], [3], [11], [12], [14]. Agile development is one of these trends; it is built on the idea that the customers can decide what they want best and whether they are getting it if software is shown to them early on and often [12]. In order to ensure fast, high quality releases, it is necessary to synchronize and automate multiple tasks from software configuration management, build management, and release management. DevOps [2] is a one of the most popular approaches that provides techniques and tools to automate software configuration management tasks needed to achieve fast and full-automated releases. There are different sets of tools that provide functionality to support DevOps approach: Serena, rPath, ServiceNow, StreamStep, UrbaneCode, etc. [2]. These tools enable full automation of build management, continuous integration, and continuous delivery processes.

In most cases, companies already have the tools and solutions needed to achieve the mentioned goals and the main challenge is to find out how to adopt and implement these tools and solutions in the new projects with minimal additional efforts (i. e., how to efficiently reuse the existing solutions). Even though the tools mentioned in [2] provide a way to fully automate a release process, there is a lack of approaches and recommendations how to reuse the same solutions in the new projects. Additional problems can be caused by refusal of the companies to use new tools; especially, if these tools are not open source or from a well-known, trusted organization. Their stance on this matter is that they might get a new problem while solving the existing ones and it is unclear whether the end result will be positive.

1.1 Problem formulation

This paper covers the following problems in the area of software configuration management:

- Use of multiple different configuration management tasks in one solution. For example, use of one 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 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.

To make configuration management-related solutions reusable, it is necessary to define a bridge from process to technical solution. First, define an abstract process of software configuration management. Then, select the abstract actions to implement general process. Finally, select and implement a concrete solution for all abstract actions. MDA (Model-Driven Approach) [9] is a popular software design approach that aims to improve the efficiency of software engineering. The same ideas have been found in the related solutions to improve reuse of software configuration management [4], [5], [6], [7], [8], [10], [13].

1.2 Novelty of paper

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 agnostic to the tools used for source code management, continuous integration, bug tracking, and build management as it only defines a way to make a solution reusable. The approach contains three levels of models to describe the configuration management process from different sides. Authors also investigate how Semantic Web technologies like OWL and SPARQL could be used to improve this approach and to perform transformations between different levels of models.

1.3 Structure

The second section introduces the approaches related to software configuration management to identify main trends. The third section provides a general description of the new approach; and the next section contains detailed descriptions of all models from the approach. The fifth section covers the use of Semantic Web technologies to perform transformation of models from different levels. Finally, the last section is related to the practical application of this approach in the field of applied software and outlines the directions for further work.

2 Related Works

The following approaches [4], [10], [5] define a configuration management process as a whole, not just a specific task. Solution in the article [4] defines a united concept of configuration management as a meta-model which allows the creation of an abstract model of software configuration. The solution is oriented to projects that use a model-driven approach. There are no recommendations on whether this approach could be used in projects that use classical development methodologies.

The principles of configuration management outlined and used to create abstract models in solution [5] are taken from the ITIL (Information Technology Infrastructure Library) standards. These abstract models allow simulation of the configuration management process and can be transformed into platform specific models later. Although this solution also includes an implementation of model-driven configuration management, it is focused on a single technology (Java).

Study [10] focuses on the integration of various configuration management tools. The maintenance of a full configuration management process requires the following tools: version control system, issue-tracking system, build server, continuous integration server, etc. Practical experience indicates that all tools work separately from each other. The main goal of the solution is to integrate different tools to solve all of the configuration management tasks. However, in order to integrate various configuration management tools together, it is necessary to define a general concept of each tool that should be integrated [10]. The study presents an ontology for software configuration management. This ontology is used as a configuration management model that describes how various configuration management tools are integrated. However, the study does not offer any instructions on how the ontology could be used for a specific project configuration management. It is not clear what kind of ontology modeling tools are advised to use and how to determine the moment when the changes have to be made.

The study [6] provides an approach to reuse installation packages in the different versions of Linux. The author outlines a problem related to the design of reusable code for installation packages. The mentioned approach is only related to Linux installation packages and does not contain any general recommendations on how to apply it for other technologies or for the other parts of configuration management.

The solution provided in [7] is oriented to increase reuse of release management with design of geographically distributed concurrent change and configuration management system. The paper describes the high-level architecture and process framework that could be used to implement new configuration management systems.

Some of the reuse-oriented configuration management approaches provide solutions only for particular tasks, for example, measurements. The article [8] presents a quantitative model for health evaluation project that helps the decision makers to make the right decision early on to amend any discrepancy that may hinder timely, high-quality software delivery.

This study is not the first attempt to introduce the Semantic Web technologies into software configuration management. In a related study, Falbo [15] proposed an SCM ontology that was used to establish a common conceptualization about the SCM domain in order to support SCM tools integration.

3 Model-driven approach for software configuration management

In the context of a new approach, the following steps are required to solve all of the software configuration management tasks and implement all the IT operations that are related to release management:

- Identify all instances where software product should be deployed. For example, TEST, QA, and PROD. Nowadays all sub-processes of general software development project usually use one particular instance. For example, DEV instance is being used for development, TEST instance is being used for testing, and PROD instance is being used by the end-users to.
- Identify all actions required to implement the flows of software changes between instances mentioned before. For example, in order to transfer changes from DEV to TEST instance, particular source code should be extracted from a source code repository, then it should be compiled to an executable file, and after that the executable file should be installed on TEST environment. Thus, the actions should be “Prepare source code”, “Build product”, and “Install product”. The actions are abstract and no details about implementation are given.
- Choose particular solutions for any abstract actions defined in the previous step. The main condition is that all solutions for all actions are stored at a centralized database. After this step, any actions can contain details about particular implementation. For example, action “Build product” has a script that builds Java project using ANT script.

A new approach has been designed according to the mentioned positions for software configuration management. The approach contains a set of different models:

- *Environment Model (EM)* – simulates all instances in a project and all flows of software changes between these instances.
- *Platform Independent Action Model (PIAM)* – simulates all tasks needed to apply all flows between the environments from Environment Model.
- *Source Code Branching Model (SCBM)* – simulates all branches of source code and directions of merge. The content of the model strongly depends on Environment Model and shows which branches should be created by version control system and defines directions for merges between the mentioned branches.
- *Platform Specific Action Model (PSAM)* – an extended variant of PIAM model where all actions are performed with specific details about implementation. All required technical details are mentioned in this model. For example: platform name, name of version control system, continuous integration server, build and installation scripts, etc.
- *Service Model (SC)* – simulates pairs of different tools from PSAM model that should be integrated with each other. To apply all actions from PSAM model, a set of different tools is required. For example, to prepare a build for a Java project, Jenkins server should have access

to Subversion version control system in order to extract source code. Therefore, Service Model should contain an element "Jenkins -> Subversion" that initializes a service that could get the information from Subversion and could post common operations from Jenkins. This server could be used by PSAM model to implement actions related to source code management.

Fig. 1 contains a general scheme of a new model-driven approach. Arrows with digits represent steps from the approach.

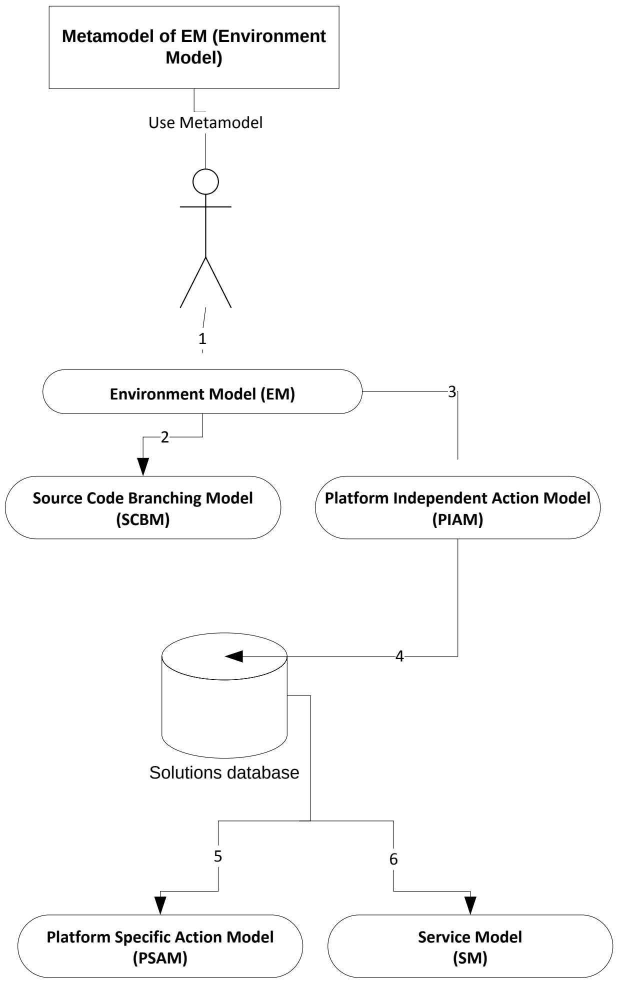
Metamodel of EM (Environment Model)
Use Metamodel
1
Environment Model (EM)
2
3
Source Code Branching Model (SCBM)
Platform Independent Action Model (PIAM)
4
Solutions database
5
6
Platform Specific Action Model (PSAM)
Service Model (SM)

Fig. 1. Model-Driven Approach for Configuration Management.

The first stage "1" represents building of Environment Model from special meta-model. Configuration manager builds Environment Model from a set of components from the mentioned meta-model. During the second stage "2", Environment Model is transformed into Source Code Branching Model. The goal of stage "3" is the transformation of Environment Model to Platform Independent Actions Model. The main task is to detect the actions needed to apply each flow between environments. The stages "5" and "6" require the interaction from a configuration manager to analyze a prepared Platform Independent Action Model and choose the solutions for each action from "Solutions Database". Structure of "Solution Database" is provided in Fig. 2.

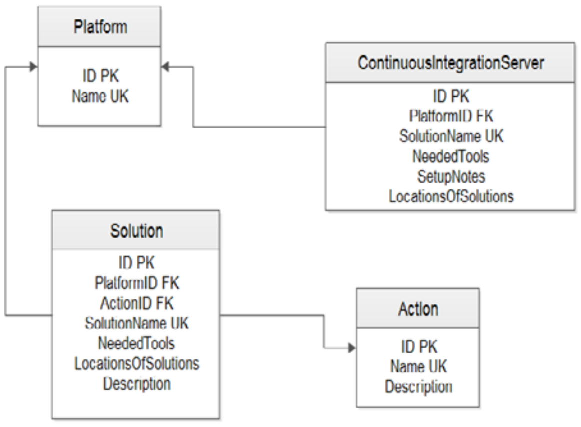

Fig. 2. Solutions Database.

Solutions Database contains all information about all configuration management actions described in PIAM model. For example, action "Compile" could have five different solutions to compile software from source code for the following technologies: Java, Ruby, C++, Oracle, C#. The mandatory requirement is that all solutions are parameterized and there are no dependencies to solutions of other actions. For example, compilation script should not know any details about other actions from PIAM, hardcodes from bug tracking management, hardcoded hosts, absolute paths etc. All details should be given as parameters. Any solution stored in Solution Database has the following attributes:

- *ID* – unique identifier in database;
- *PlatformID* – reference to platform;
- *ActionID* – reference to action. Table "Action" contains all possible actions from PIAM meta-model;
- *NeededTools* – set of tools to implement current solution;
- *LocationsOfSolutions* – information about ready scripts, frameworks, functions, including paths, locations of servers, web-pages etc.;
- *Description* – some notes provide additional information about implementation.

3 Models for software configuration management

According to the model-driven approach provided in the previous section of this paper, the following models should be used to describe an implementation of software configuration management process.

3.1 Environment Model

Environment Model simulates developers, instances, and flows of changes between the mentioned instances. The key element of Environment Model is Environment. In the context of EM, Environment is an infrastructure (i. e., servers, applications, web-services, etc.) for particular process. For example, DEV environment is for development and TEST environment is for testing. Environment Model shows the flows of changes between different environments. From configuration management side, it is important to detect a way how particular changes have been made.

The following attributes are defined to describe Environment in the context of EM model:

- Name – unique name of Environment;
- Description – some notes about scope of the current Environment;
- CustomerSupportFlag – flag that defines whether this environment is supported by customer;
- DevelopmentFlag – flag that defines whether this environment is for development;
- OriginalEnvironmentFlag – flag that defines a scope of the environment: sub-process of project (development, testing, quality accepting) or only integration or testing builds;
- OriginalEnvironmentName – In case when OriginalEnvironmentFlag attribute value is set to “false”, this attribute should contain a name of the corresponding original environment. For example, if current environment is a copy of the original test environment, this attribute should contain a name of the original test environment.

Example of Environment Model is given in Fig.3.

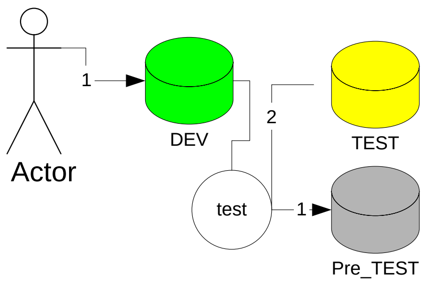

Fig. 3. Environment Model.

3.2 Platform Independent Action Model

The purpose of Platform Independent Action Model is a simulation of abstract actions needed to apply all flows from Environment Model. PIAM model has the following elements:

- ContinuousIntegrationServer – simulates a framework for implementation of configuration management actions. This element has the following attributes:
  - PlatformName – information about platform,
  - SolutionName – unique name,
  - NeededTools – tools needed to implement the current framework,
  - SetupNotes – information about installation,
  - LocationsOfSolutions – location of complete scripts or frameworks.
- Abstract actions that simulate sub-tasks of general process:
  - DEVELOPMENT – simulates development by programmers and regulations that are used to control quality of development.
  - COMMIT_CHANGES – simulates submission of changes (i. e., commit operation) to a version control system.
  - PREPARE_BASELINE – simulates source code management actions that involve operations with branches, baselines, and transfers of changes between different branches.
  - BUILD_PRODUCT – simulates build management and all operations related to building executables from particular source codes.
  - INSTALL_PRODUCT – simulates all actions related to installation of particular builds.
  - DELIVERY_PRODUCT – simulates preparation of an installation package for software product.
  - ENV_UPDATE_NOTIFICATION – simulates all actions related to status management. After a customer updates the

environment, a supplier should apply a set of operations to confirm the update (change statuses in a bug tracking system, refresh a promotion branch, send notifications to a project team, etc.).

- Events – all events from Environment model.

3.3 Source Code Branching Model

The purpose of Source Code Branching Model is to define a general strategy on how to manage source code according to Environment Model. In general, this model shows which branches are needed to support a code baseline for all original environments from Environment Model. Additionally, SCBM model defines directions of merges between different branches.

3.4 Service Model

Service Model shows a set of tools, which should be able to call from continuous integration server defined at Platform Independent Action Model. Service Detection Algorithm takes all actions from PSAM model, extracts the value of NeededTools attribute, and prepares the following pairs:

ContinuousIntegrationServer.NeededTools: Action(i).NeededTools,

where i – number of action from PSAM, 0 < i < General Count of Actions.

These pairs of tools define that it is needed to develop a set of services to execute remotely common functions of tools in attribute "NeededTools" from continuous integration server before PSAM model will be implemented.

4 Model transformation using Semantic Web technologies

The Semantic Web 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. Authors think that the Semantic Web technologies can be efficiently utilized in the Software Configuration Management (SCM) 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 [16].

According to Guarino, ontologies can be classified into the following categories [17]: (1) foundational ontologies (also called top-level ontologies), which describe very general concepts such as time, object, event, action, etc., (2) domain ontologies, which describe the conceptualization related to a generic domain (for example, finance, medicine, etc.), (3) task ontologies, which describe the conceptualization related to a generic task (for example, sale), and (4) application ontologies, which describe concepts dependent on a particular domain and task.

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.

Arantes et al [17] came to the conclusion that ontology from [15] 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 model-driven approach. That would allow the reuse of the expert knowledge encapsulated within this ontology. However, it is necessary to correspondingly modify it according to the needs of this approach. One of the main reasons for these changes is that the ontology was not designed with a good reasoning

support. However, reasoning plays a very important role in the ontology-based model transformation.
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 [18]. 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.
OWL does not restrain developers to use any particular inference engine (i. e., they are free to choose what they want to use). However, the standard reasoners like Pellet or FACT++ have a few major disadvantages: they have a limited functionality (i. e., some rules can’t be evaluated with the existing open-source reasoners) and their performance is not that good. Therefore, in order to provide extra functionality it might be needed to use additional tools (for example, Jena's library or some of the Protege plugins).
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 [19]. 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 $a_1 \wedge ... \wedge a_n$. 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 [20]. This experience suggests that SWRL rules can also be used to implement ontology-based SCM model transformation rules.
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.

4 Use cases of model-driven approach in applied software fields

The proposed methodology of software configuration management based on model-driven approach and semantic web has been validated in the process of development of Web-based business oriented portals in the areas of e-logistics.

Several business portals in e-logistics were developed within the frameworks of different EU funded projects: eINTERASIA [23],eLogmar-M and eLogmar.eu [22]

Each next Web portals version is based on the results of previous projects and MDA and Semantic Web approach (Fig. 4).

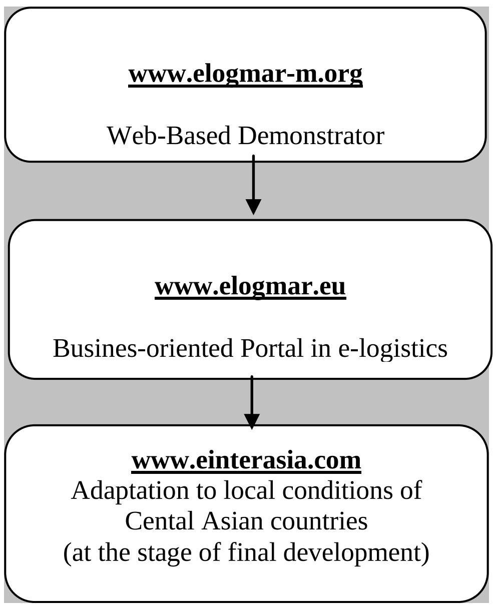
www.elogmar-m.org
Web-Based Demonstrator
www.elogmar.eu
Busines-oriented Portal in e-logistics
www.einterasia.com
Adaptation to local conditions of Cental Asian countries
(at the stage of final development)

Fig. 4. The results of EU funded projects.

The main goal of the eLogmar-M project [22] was to create a Web portal demonstrator which incorporates business process of different partners operation along the selected maritime freight route. The demonstrator provides online cooperation for two target groups.

The following business processes are supported by the Web portal:

1. Cargo group:
   - Preparation of a contract for Sale/ Purchase
   - Looking for suitable actors from cargo transportation group in accordance with the terms and conditions from a contract of Sale/ Purchase and estimation of a start-to-finish transportation rate.
2. Transportation group:
   - Promotion of services by the way of information via Internet and mobile communication
   - Preparation and maintenance of the initial information about services.

www.elogmar.eu Web portal represents commercialized, business-oriented version of the demonstrator produced by eLogmar-M project [22].

Additional functionalities were implemented:

- Integration of PayPal payment system
- Modification of Web interface forms taking into account the results of demonstrator’s validation
- Improvement of homepage design etc.

The 7th Framework Program project eINTERAISA [23] is aimed at the adaptation, dissemination, and local exploitation of EU research results in Central Asian countries. The development of an integrated Web portal in the area of e-logistics adapted to the business requirements and specific needs of the target region is one of the main objectives of the project.

It incorporates functionalities of www.elogmar.eu portal, cargo auction mode and research results of selected EU projects.

Application of model- and Semantic Web -based approach showed its ability to support the development and configuration management processes of applied software.

5 Conclusions

The paper presents a new model-driven approach for implementation of software configuration management process. Unlike other related approaches, it describes a whole development process: from planning to technical implementation. The approach is oriented to increase reuse of existing solutions using well-known and trusted tools. A set of meta-models was designed to describe software configuration management process using models. Authors also investigate how Semantic Web technologies like OWL and SPARQL could be used to improve this approach and to perform transformations between different levels of models. Finally, validation of new methodology described.

The most important further work is the development of a tool to automate process of creating and transforming mentioned models. A set of experiments will be performed in order to evaluate the effectiveness of the new model-driven approach. Authors hope that the results of the experiments will contain feedback that could be used to improve models in the approach.

Actually, model-driven approach is abstract and initially it shows only steps, types of models, and relations between them. It means that implementation of the models could be different from the one provided in this paper. It could generate new ideas on how to improve the existing models or how to implement it in another way.

Acknowledgments

This work is partly funded by Latvian State Research Program „Cyber-physical systems, ontologies and biophotonics for safe&smart city and society”.

References:

[1] Bill Chamberlin's HorizonWatching. 2014. Top 18 Trends in Application Software Development for 2014 | Bill Chamberlin's HorizonWatching. [ONLINE] Available at: http://www.billchamberlin.com/top-18-trends-in-application-software-development-for-2014/. [Accessed 20 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 [Accessed 20 October 2014].

[3] 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. [Accessed 20 October 2014].

[4] 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.

[5] 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.

[6] Guo., P.,J., CDE: A Tool for Creating Portable Experimental Software Packages. Computing in Science & Engineering (Volume:14 , Issue: 4), Pages: 32-35, 2012.

[7] Munirul, I., CCMS: A geographically distributed concurrent change and configuration management system, Bell Labs Technical Journal (Volume:8 , Issue: 3), 2014.

[8] Nagy, A., A Bayesian Based Method for Agile Software Development Release Planning and Project Health Monitoring, Intelligent Networking and Collaborative Systems (INCOS), 2010.

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

[10] Pindhofer W., Model Driven Configuration Management. Master work of Wien University, Wien, 2009.

[11] Taking Release Management to the Next Level. 2014. [ONLINE] Available at: http://www.slideshare.net/xebialabs/taking-releasemanagementtothenextlevel. [Accessed 20 October 2014].

[12] Trends in Software Engineering - Dice News. 2014. [ONLINE] Available at: http://news.dice.com/software-engineering-talent-community/trends/. [Accessed 20 October 2014].

[13] van der Storm, T., The Sisyphus Continuous Integration System, Software Maintenance and Reengineering, 2007. CSMR '07., 2007.

[14] 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/. [Accessed 20 October 2014].

[15] 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.

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

[17] Guarino, N. Formal Ontology and Information Systems. In: Formal Ontologies in Information Suystems, IOS Press, 1998, 3 - 15.

[18] 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

[19] Dalwadi, N. et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (3) , 2012,pp. 3843-3847.

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

[21] 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.

[22] Novickis L., Vinichenko S. Essential Logistics Principles for Creating a Web-Portal of Transport Services Consumers. In: Scientific Proceedings of the eLOGMAR-M Project. It &T Solutions in Logistics and Maritime Applications, ISBN: 9984-30-119-2, Riga, JUMI Printed House, 2006, pp.21-29.

[23] ICT Transfer Concept for Adaptation, Dissemination and Local Exploitation of European Research Results in Central Asian Countries, 2013. {ONLINE} Available at http:// www.einterasia.eu {Accessed 26 October 2014].