Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 43 (2015) 3 – 10

Procedia Computer Science

ICTE in Regional Development, December 2014, Valmiera, Latvia

Models for Implementation of Software Configuration Management

Arturs Bartusevics[a]* Leonids Novickis[a]

[a]Faculty of Computer Science and Information Technology, RTU, Kalku Street 1, Riga LV-1658, Latvia

---

Abstract

The main scope of software configuration management is control of software evolution process to include at final version only valid and tested items. To achieve this, software configuration management have to prepare solutions for tasks such as identification of software configuration items, version control, build and deploy management etc. The paper provides new model-driven approach for implementation of software configuration management. New approach is supported by set of models to describe software configuration management process from different sides. New approach helps to organize existing solutions in parameterized way that increase ability of its reuse. Current paper introduces to problems in software configuration management area and main trends of new solutions. After introduction, new model-driven approach described. The second part of paper provides models for new approach. Presentation of models combined with simplified use case, which illustrates practical application of models. Finally, directions of further researches are provided.

Keywords: Software Configuration Management; Model-Driven Approach.

---

1. Introduction

During last few years iterative methodologies for software development projects become more popular. A time moment needed to develop ready software product is so long, that customers would like to see intermediate results to be sure that product agrees with initial requirements[1, 2, 3, 4]. Iterative methodologies like Agile requires often releases of ready product to support ability of testing quality during continuous development. Business would like to get new version of product as soon as possible and related IT operations have to support it[3]. But great speed of creating new

---

* Corresponding author.
E-mail address: arturik16@inbox.lv

doi:10.1016/j.procs.2014.12.002

releases also requires high quality of items, included at mentioned release. Software configuration management is a discipline that controls evolution of software product and allows including only valid, expected and tested items to final version. So, main tasks of software configuration management, such as identification of configuration items, version control, status accounting, should be united with build management and release management to provide valid and often releases[3,4].

The increase quality and speed of new releases, different approaches could be used. There are some different trends of solutions related to software configuration management.

1.1. Problem

- Lack of methodologies oriented to development of reuse oriented solutions for software configuration management with existing and well-knows tools;
- Instead of huge amount of tools and standards related to software configuration management, there is lack of approaches that can show a way from abstract view of overall process to concrete technical solutions. As a result, software configuration management is a set of practices, week engineering requires additional resources to fix different errors.

1.2. Scientific novelty

The study provides new model-driven approach for implementation of software configuration management. Unlike other approaches, it is not oriented to particular tool that “should solve any problem” but provides the steps how to increase the reuse of existing solutions. Well-known tools for source code management, continuous integration, bug tracking and build management could be used, but provided approach shows only a way how to achieve reuse of solutions. New approach contains three levels of models to describe configuration management process from different sides. Models and relations between them provide a way from the general process overview to concrete technical solutions.

2. Related works

As far back as 1992 there was published an article[5] introduced to main challenges of configuration management area. One of the main ideas is related to development of service model for configuration management process. Many things have changed since then; more standards are developed in software development area, new tools for configuration management are designed. In a recent interview with a long-term expert in configuration management area[6] was mentioned the year 1998, when there was an attempt to create a “super tool” for integration of all solutions of configuration management in one framework. Attempt was failed, because solutions was too complicated. Configuration managers and developers were afraid of "majesty" of such tool. Configuration management expert[6] emphasizes challenge to enhance trust between configuration managers and programmers as the main future challenge. The main requirement for this is a clear procedure, which could be trusted by developers. Other configuration management experts[1, 2] note that solutions will be ineffective and will require additional resources without planning of general process before implementation of particular solutions and installation of tools. Modern solutions require reusable approaches that allow coming efficiently from the process general requirements to technical implementation.

During analysing different approaches of reuse oriented solutions, more ideas from MDA[7] have been found. The important task in configuration management is the source code management and significant part of model-driven solutions is related to this task[8, 9, 10]. New approaches try to improve source code management by modelling of product components, streamlines and branches[10]. Abstract models designed to improve new development of source code management systems[8, 9]. There are solutions provide an abstract model for general configuration management process based on software quality standards[11, 12, 13]. Usually the approaches do not provided how to increase reuse of existing solutions. It could be very important because software development companies usually have a set of concrete tools that are trusted from theirs point of view. So, new tools or methods with “super performance”, “mystic full-automated feature” could not be trusted and acceptable by companies. The following works[7, 14, 15] consider software configuration management process as a whole, not just a particular task. Approach from article[16] provides general concept of configuration management and meta-model for creating different models of software configuration. The solution is focused on projects where development is based on model-driven approach, but there are no explanations how this approach could be used in projects with other development approaches. The main concept of configuration management in study[15] was taken from the ITIL (Information Technology Infrastructure Library) standards abstract model was designed. Later this model could be transformed to platform specific model. Although that solution also includes an implementation for model-driven configuration management, it is focused on a single technology (JAVA). No any recommendations are provided how to integrate together different tasks of configuration management such as source code management, build management and release management.

Study[14] focuses on various mutual integration of configuration management different tools. 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 etc. New model-driven approach provided in current paper, supports the main ideas described in related works about models. Unlike related works, models in provided solution have strong defined connections between each other and provides full way from abstract process overview to concrete implementation of particular tools, scripts or frameworks. This could reduce efforts for invalid customizations of technical solutions. Additionally, new approach is not oriented to any specific tool but allows to use existing, well-known and trusted. The approach provides only a way how to refactor existing solutions and design new one to increase its reuse. This could save up time for implementation a similar solution for other projects.

3. General approach for implementation of software configuration management

During design of new approach for implementation of software configuration management, new position for interpretation of mentioned process has been defined. To solve all tasks related to software configuration management and implement all IT operations related to release management, the following steps are required:

- Identify all instances where software product should be released, for example, TEST, QA, and PROD. Nowadays all sub-process of general software development project usually use particular instance[1, 2]. For example, DEV instance using for development, TEST instance for testing, but users working with ready product in PROD instance;
- Identify all actions required to implement all flows of software changes between instances mentioned before. For example, to move changes from DEV to TEST instance, particular source code should be extracted from source code repository; it should be compiled to executable file and after executable file should be installed on TEST instance. So, actions should be “Prepare source code”, “Build product” and “Install product”. Actions are abstract and no any details about implementation are given;
- Choose particular solutions for any abstract actions defined at previous step. The main condition is that all solutions for all actions are stored at centralized database. After this step, any action have details about implementation. For example, actions “Build product” has particular script that builds JAVA project by ANT script.

According to mentioned position for software configuration management, new approach has been designed. Approach contains a set of different models:

- Environment Model (EM) – simulates all instances in project and all flows of software changes between mentioned instances.
- Platform Independent Action Model (PIAM) – simulates all tasks needed to apply all flows between instances from Environment Model.
- Source Code Branching Model (SCBM) – simulates all branches of source code and merging directions. The content of model strongly dependent from Environment Model and shows which branches should be created at version control system and defines directions of merges between mentioned branches.

- Platform Specific Action Model (PSAM) – extended variant of PIAM model where all actions are fulfilled with specific details about implementation. In this model all needed technical details are mentioned, 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 are required. For example, to prepare build for JAVA project, Jenkins server should have access to Subversion version control system to extract source code. So, Service Model should contains an element "Jenkins -> Subversion" that initialize service that could get 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.

General picture of new model-driven approach is given on Fig. 1. Arrows with digits means steps of approach.

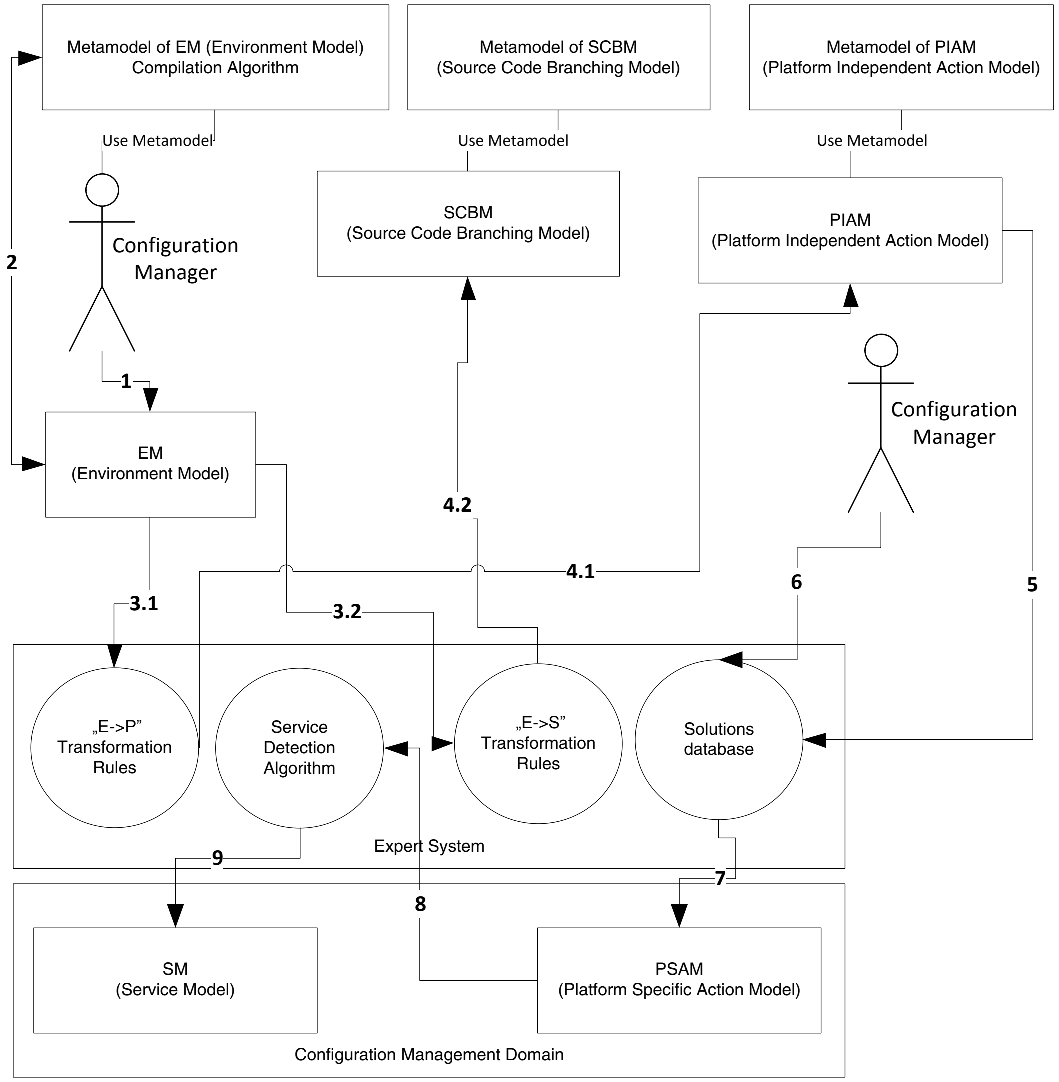
Metamodel of EM (Environment Model)
Compilation Algorithm
Metamodel of SCBM
(Source Code Branching Model)
Metamodel of PIAM
(Platform Independent Action Model)
Use Metamodel
Use Metamodel
Use Metamodel
Configuration Manager
SCBM
(Source Code Branching Model)
PIAM
(Platform Independent Action Model)
2
1
EM
(Environment Model)
Configuration Manager
4.2
4.1
6
5
3.1
3.2
„E->P" Transformation Rules
Service Detection Algorithm
„E->S" Transformation Rules
Solutions database
Expert System
9
8
7
SM
(Service Model)
PSAM
(Platform Specific Action Model)
Configuration Management Domain

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

New model-driven approach contains the meta-models for EM, SCBM and PIAM models. All mentioned models are dependent from each other. Model-driven approach contains special element called “Expert System”. In context of this work Expert System is a set of special blocks of rules. This rules define dependencies between different models. Using these rules, one model could be created automatically from other model. The following blocks defined in “Expert System”:

- “E->P” – rules that define how to create Platform Independent Action Model from Environment Model. In left side of each rule particular condition of elements from Environment Model is defined. A right side of each rule contains a set of actions from PIAM meta-model needed to apply particular flow from Environment Model. For example, if Environment Model contains a flow of changes between DEV and TEST environment, the following abstract actions should be selected: prepare source code for TEST environment, build product from mentioned source code, install build to TEST environment.
- “E->S” – rules that define branches at source code management system and directions of merges depends on instances from Environment Model. A left side of each rule contains state of instances from Environment Model, but a right side contains a set of branches and directions of merges. As a result, source code management and branching strategy could be selected according to instances of particular project.
- “Service Detection Algorithm” – detects all pairs of tools from PSAM model that have to be integrated together to apply exchange of information and post common methods to other tool, for example commit changes to Subversion repository from Jenkins script.

A model-driven approach (see Fig. 1.) at the first step “1” requires interaction from configuration manager. Configuration manager creates Environment Model from particular meta-model. Meta-model for EM contains a compilation algorithm and during step “2” it compiles a model created by configuration manager. Compiled Environment Model should be sent to block “E->P” and “E->S” during steps “3.1” and “3.2”. As a result, transformation rules in mentioned block and meta-models of PIAM and SCBM create Platform independent Model and Source Code Branching Model. This actions are marked as steps “4.1” and “4.2”. After step “4.2” the second manual interaction from configuration manager is required. Hi should select solution for each action in PIAM model from “Solution 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 does not have dependencies from solutions of other actions. During steps “5” and “6” configuration manager should select solution for any action from PIAM model. As a result, extended variant of PIAM should be created, called Platform Specific Action Model. This model should be ready after step “7” is complete. Last steps of model-driven approach marked as “8” and “9” prepare Service Model from PSAM. Service Model shows all services for each tool that could be called from continuous integration server. For example, if PSAM model contains Jenkins server for continuous integration and Subversion for source code management, Service Model will define service “Jenkins -> Subversion”. This service should be called from Jenkins to get any attributes form Subversion (details about commits) and to post common commands (merge, update, commit). Finally, to implement software configuration management process by provided approach, Service Model and Platform Specific Action Model should be implemented at configuration management domain (see Fig. 1.).

4. Models of software configuration management

To implement approach for software configuration management provided at previous section, a set of models has been designed to describe software configuration management process.

4.1. Environment Model (EM)

The scope of Environment Model is simulation of developers, instances, and flows of software changes between mentioned instances. The main element of Environment Model is Environment. Environment, in context of EM, is an infrastructure (servers, applications, web-services etc.) for particular process. For example, DEV environment is for development, TEST environment is for testing and PROD environment is for software exploitation. Environment Model shows flows of changes between different environments. From configuration management side, it is quite important to detect a way how to particular changes have been made. According to this, the following kinds of environments are defined: Development Environment, Original Environment and Customer Support Environment.

Environment Model has three kinds of elements: Environment, Event and Actor. Implementation of Environment Model starts with creation of elements and relations between. After, visual elements of Environment Model should be transformed to XML format. Environment Model in XML format should be transferred to meta-model of Environment Model, which compiles it by special compilation algorithm (see Fig. 1., step "2"). As a result, configuration manager has compiled Environment Model ready to further use.

4.2. Platform Independent Action Model (PIAM) and Platform Specific Action Model (PSAM)

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

- ContinuousIntegrationServer – simulates a framework for implementation of configuration management actions, because all actions of process should be independent from particular workstation of configuration manager[1]. This element has a set of attributes:
  - PlatformName – name of platform,
  - SolutionName – unique name,
  - NeededTools – tools needed for implementation of current framework,
  - SetupNotes – useful information about installation steps,
  - LocationsOfSolutions – location of ready scripts, frameworks etc., that could be used during implementation.
- Abstract actions that simulates sub-tasks of general software configuration management process:
  - DEVELOPMENT – simulates development by programmers, frameworks and regulations to control quality of developed changes.
  - COMMIT_CHANGES – simulates action related to save developed changes to version control system. This could provide user guide how to save changes in version control repository to support general requirements for specific quality procedure.
  - PREPARE_BASELINE – simulates source code management action related to operations with branches, baselines and transfers of changes between different trees of source code. This could provide scripts, frameworks etc., to organize common source code management operations.
  - BUILD_PRODUCT – simulates build management and all operations related to building executables from particular source code.
  - INSTALL_PRODUCT – simulates all actions related to installation of particular builds to environment.
  - DELIVERY_PRODUCT – simulates preparing of installation package for ready software product. Installation package should be prepared for environments that are supported by customer.
  - ENV_UPDATE_NOTIFICATION – simulates all actions related to status accounting of software changes. After update of environment, which is supported by customer, supplier should apply a set of operations to fix update fact (change statuses in bug tracking system, refresh promotion branch, send notifications etc.).
- Events – all events from Environment model.

During steps "3.1" and "4.1" of model-driven approach (see Fig. 1.), PIAM model preparing from Environment Model by "E->P" transformation rules. "E->P" transformation rules defines which abstract actions are required to apply particular flow of changes between environments. On the left side of rules (IF) are conditions of attributes of environments between them particular flow should by applied, but on the right side (THEN) is a set of needed abstract actions from PIAM. According to this, preparation of Platform Independent Action Model from Environment Model by "E->P" contains the following steps:

- Create empty PIAM model;
- Copy all Events from EM to PIAM;

- Apply transformation rules to explore conditions of all environments related to particular Events and define abstract actions for each flow of each Event;
- Attributes of element ContinuousIntegrationServer are empty in PIAM model.

PIAM model contains only a set of abstract actions needed to apply each flow of each Event from Environment Model. During steps “5” and “6” (see Fig. 1.), configuration manager should select solution from Solution Database for each action. As a result, Platform Specific Action Model should be prepared as extended variant of PIAM. PSAM model has the same elements, but attributes of continuous integration server element are filled by values from Solution Database. Solutions for continuous integration server and each abstract action should be selected from Solution Database during creation of PSAM model. In this example continuous integration server is Jenkins which should be installed on Linux platform. Additionally, concrete solutions should be selected from Solution Database for the following actions: PREPARE_BASELINE, BUILD_PRODUCT, INSTALL_PRODUCT.

4.3. Source Code Branching Model (SCBM)

The main task of Source Code Branching Model is detection of a general strategy how to manage source code of product according to environments. 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. Example of SCBM model is provided (see Fig. 2.). The model is created from Environment Model by transformation rules “E->S” during steps “3.2.” and “4.2.” (see Fig. 1.).

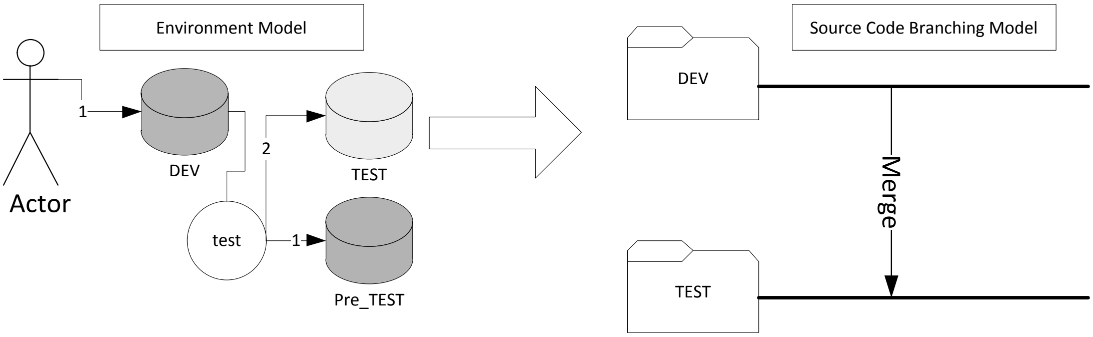

Fig. 2. Environment Model and Source Code Branching Model.

Example of SCBM model shows that branches should be created for each original environment (DEV, TEST) to support actual status of source code. Directions of merges between environments show how to move changes of source code from one branch to other.

4.4. Service Model (SM)

Service Model shows a set of tools which should be able from continuous integration server defined at PSAM model. Service Detection Algorithm (see Fig. 1.) takes all actions from PSAM model, extract attribute NeededTools. Before implementation of PSAM model, two services should be developed. The first service should provide a framework that allows calling any common action of Subversion system (update, commit, merge, etc.) from Jenkins server. Service, for example, could be a Linux shell script with functions “commit”, “update” “merge”. During development of mentioned service, the fact, that Jenkins is installed on Linux platform, should be taken in account. The second service should allow making any actions, related to build executable on Oracle WebLogic Server. Only after mentioned services are developed, PSAM model could be implemented at configuration management domain.

5. Conclusions and further works

The paper provides introduction to new model-driven approach for implementation of software configuration management process. Unlike other related approaches, new one provides a way from planning to technical implementation, do not impose to use particular tools and is oriented to increase reuse of existing solutions using well-known and trusted tools. To describe software configuration management process by models, a set of meta-models and transformation rules are designed. Using simplified use case, models are illustrated in current article. The most important of further works is development of tool to automate process of creating and transforming mentioned models. A set of experiments will be planned to fix gains from new model-driven approach. Author's hope that results of experiments will generates novel ideas related to improve models of provided approach. Actually, provided model-driven approach is abstract and initially it shows only steps, kinds of models and relations between them. It means that implementation of the models could be not the same as provided in this paper. It could generate new ideas how to improve existing models or how to implement its by other way.

References

1. Aiello, R., Configuration Management Best Practices: Practical Methods that Work in the Real World (1st ed.). Addison-Wesley, 2010.
2. Berczuk, A., Software Configuration Management Patterns: Effective TeamWork, Practical Integration (1st ed.). Addison-Wesley, 2003.
3. Home - DevOps.comDevOps.com | Where the world meets DevOps. 2014. [ONLINE] Available at: http://devops.com/. [Accessed 13 October 2014].
4. Удовиченко, Ю. Управление изменениями и кессонная болезнь проектов. Available at: http://experience.openquality.ru/software-configuration-management/, 2011.
5. Dart, S., The Past, Present, and Future of Configuration Management. CMU/SEI-92-TR-8, 1, 25., 1992.
6. 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].
7. Nikiforova O., Pavlova N., Gusarovs K., Gorbiks O., Vorotilovs J., Zaharovs A., Umanovskis D., Sejans J. Development of the Tool for Transformation of The Two-Hemisphere Model to The UML Class Diagram: Tehnical Solutions and Lessons Learned. Proceedings of the 5-th International Scientific Conference „Applied Information and Communication Tehnologies", 2012, Jelgava, Latvia, pp. 11-19.
8. 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.
9. 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.
10. Toth, Z., 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., 2013.
11. Estublier, J., 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., 2000.
12. Ruan, Li., 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., 2003.
13. Mingzhi, M., 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., 2008.
14. 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.
15. 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.
16. Pindhofer W., Model Driven Configuration Management. Master work of Wien University, Wien, 2009.
17. Osis J., Asnina E., Model-Driven Domain Analysis and Software Development: Architectures and Functions. IGI Global, Hershey - New York, 2011, 514 p.