doi: 10.2478/v10313-012-0010-y
2012 /15

A Method for the Effective Configuration of Reuse-Oriented Software Release and Its Application in the Field of Insurance

Arturs Bartusevics[1], Vladimirs Kotovs[2], Leonids Novickis[3], [1-3]*Riga Technical University*

***Abstract* – The paper presents the results of the research related to the improvement of software configuration and release management processes for software maintenance projects. The study describes a simple, effective and reusable solution to build high-quality releases that include not all, but only tested changes. The proposed method produces the list of components to be included into the release build based on the list of changes planned for the upcoming release and information from the project code version control and issue tracking systems.**

***Keywords* – software configuration management, reuse-oriented process, insurance business processes**

I. Introduction

Nowadays with the growing demand for rapid software system development, researchers are continuously looking for new software engineering methodologies and techniques in order to improve software development processes, to ensure high quality of software products and to make its maintenance and support easier.

The need for systematic reusable approaches and effective software process methods, which allow addressing recurring problems successfully, is obvious and important. A process in software engineering could be defined as a set of activities that lead to the production of a software product [1], and it is important in order to ensure efficiency, reproducibility, homogeneity, and predictable time and effort constraints.

The study concerns improvements in software configuration and release management processes that address the tasks of tracking and controlling changes in the software and preparing the product for deployment to its users. Being fine-grained parts of the process dimension within the reuse-oriented framework described by us in [2], the effective reuse-oriented release and configuration management processes assume proper development of

- managerial infrastructure in the form of a set of functions, responsibilities, reporting requirements, and reward, required to ensure proper operation of the processes, and
- technological infrastructure to support operations and enforcement of testing, verification and other standards.

The main objective of the study is to identify and implement a simple, effective and reusable solution to build high-quality releases that include not all, but only tested changes. The topic is of particular significance since the proposed approach may provide some economic and organizational benefits a) by increasing the quality of product releases, b) by reducing development and operating costs, and c) by efficient utilization of development knowledge and corporate expertise.

The following section identifies the general prerequisites for the application of the proposed method that is followed by the detailed review of the proposed release building method and discussion of its proof-of-concept implementation, current limitations and application in the field of insurance. Finally, the conclusions are made and future work is outlined in the last section.

II. General Prerequisites for the Method

The paper discusses the method that aims to improve software configuration and release management processes for software maintenance projects, where the product is a large information system being maintained using one of the iterative development methodologies. The maintenance phase of a large project often assumes the existence of issue tracking system, so that the project management could effectively plan which defects to be resolved and change requests implemented for the particular system release. Unfortunately, it is a rather common situation in practice, when there are multiple unsolved or not tested changes on the planned release date that could not be delivered with the release. As a result, the developers may face the challenge to prepare a product release that includes not all, but only the components with all tested changes.

The following general prerequisites considered important for the application of the proposed method are described in Table I. It should be noted that all these requirements are usually covered by the configuration management or quality assurance standards adapted by the organization, and there are many commercial and non-commercial tools available to help resolving these prerequisites efficiently.

However, for the purpose of our research is not enough just to introduce the defect tracking and source version control systems. There is a need to link these two systems together. Namely, the change identifier from the issue tracking system should be defined for any modification in the source code and propagated to the version control system. It is important to guarantee the control for this action, so that it would be impossible to commit modifications to the version control system without specifying the change identifier.

TABLE I

GENERAL PREREQUISITES FOR THE METHOD

Prerequisite	Description
Defect and change request tracking process	Every problem or change request should be registered and assigned a unique identifier using the centralized issue tracking system that makes it possible to manage and monitor the status of every change over time.
Source code version control	It is necessary to effectively manage changes in the product source code using version control systems. There should be information available about all changes ever made in every unit of source code.
Continuous integration process	It is advisable to adapt a continuous integration process to automatically build a product from certain source code changes, to make sure that the build is successful and meets certain quality control criteria by successful execution of integration and unit test suites.

For example, Subversion VCS (version control system) allows you to create a trigger that commits changes to a centralized server and verifies if the description contains at least one identifier and whether it actually exists in the issue tracking system.

Additional requirement for the application of the proposed approach concerns the technologies adhered to a particular product. In general, the technology stack used for development might either allow building single product elements independently, or require building a complete product from all its elements. As examples of these two product types, one might consider the database application of Oracle Forms, where it is usually sufficient to perform one or more scripts for changes to be implemented, and Ruby on Rails project, where a change requires rebuilding the entire system from all source code files. The proposed method is supposed to be applied to the projects of the first type, where it is possible to construct single product elements independently.

III. THE METHOD FOR EFFECTIVE SOFTWARE RELEASE CONFIGURATION

To apply the method for building high-quality releases that include only tested changes, two conceptual tools are defined:

a) Analyzer – performs the analysis of the product source code in the version control system in order to classify its elements by issues related to committed changes
b) ReleaseBuilder – performs the project build from a defined set of source elements and generates installation package or script based on the Analyzer results. The following structural elements constitute the core of ReleaseBuilder:
   - Description of the project source code file types (e.g. database scripts, Oracle Forms files, XML files, etc)
   - Definition of installation functions for each type of source code file
   - Metadata about the repository structure used to analyze project source code files and to determine proper installation functions.

Figure 1 outlines the following major steps of the method proposed to be scheduled as tasks on a continuous integration server:

1. Receive a list of issues planned for inclusion in the product release
2. Launch Analyzer to determine the related change identifiers in the version control system
3. Launch ReleaseBuilder to generate an installation script or package
4. Install the build to test the instance
5. Run automatic integration and unit tests and prepare deployment artifacts for delivery.

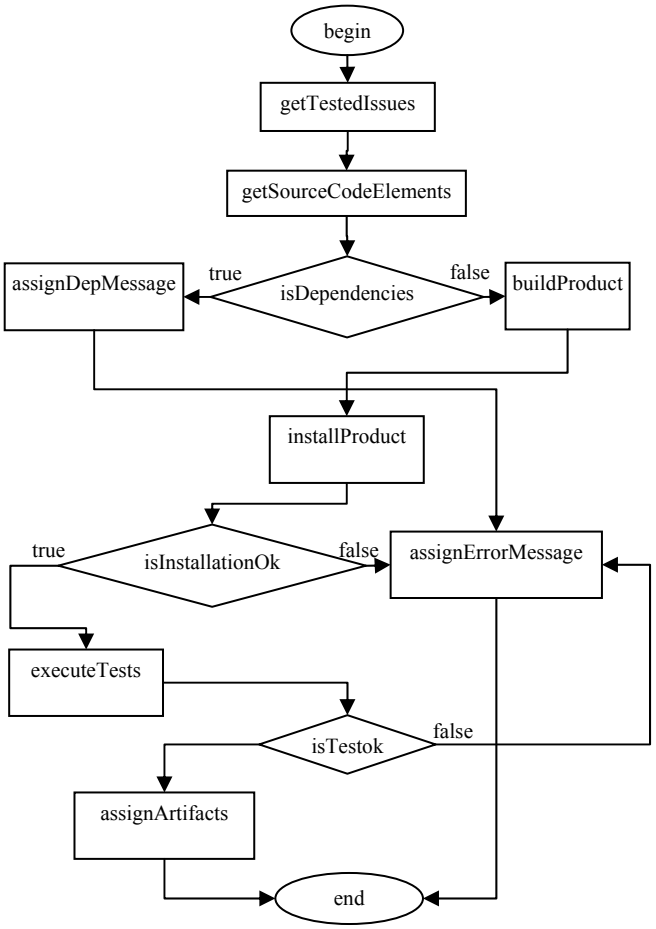

Fig. 1. The major steps of the method

- getTestedIssues – connects to the issue tracking system and selects issues according to the supplied criteria
- getSourceCodeElements – searches for the source code elements related to the selected issues
- isDependencies – verifies if the selected source code elements have dependencies on other parts related to issues not tested yet
- buildProduct – produces an installation script from the provided list of VCS element identifiers
- installProduct – deploys the product build to test, runs the prepared installation script on a test instance

- isInstallationOk – verifies if the installation on continuous integration server is successful
- executeTests – runs integration and unit test suites on the test instance to make sure the build is successful and meets certain quality control criteria
- isTestOk – verifies whether the execution of integration and unit test suites on a continuous integration server is successful
- assignArtifacts – prepares product deployment artifacts

IV. Method Implementation Details

The Analyzer and ReleaseBuilder tools have been developed as Java utility applications for the proof-of-concept implementation of the defined release configuration method. The applications are designed to be integrated with Subversion code repository and are easily adaptable to practically any continuous integration server.

A. Analyzer

The application is an executable JAR file that relies on the functionality provided by SVNKIT library [3] that is specially designed to expand opportunities for administration of Subversion repositories. The application receives the following input parameters:

- Repository connection details
- List of issues planned for inclusion to the release.

As mentioned previously, Subversion VCS assigns a unique revision identifier for each commit of modifications in the source code tree, and it is possible to describe the revision by referencing issue numbers from the defect and issue tracking system in the description of commit.

The algorithm performs the search for the revisions related to the issues planned for inclusion into the release by analyzing descriptions of commits. Based on this data, the list of modified and added files is derived, and the revision number of previous modification is found for each file from the list. If the previous revision of the file is not included into any of previous project releases, the following information is persisted into a special data structure:

A. VCS revision number;
B. Files added, modified or deleted in the revision;
C. Identifier of the issue related to the revision.

When the processing of each file is completed, the information about the found dependencies is persisted for further analysis. If no dependency is found, the product can be built successfully from only the tested changes. In this case, the revisions found are saved in a file for processing by ReleaseBuilder to prepare an installation package or script. Figure 2 outlines the major steps of Analyzer workflow.

B. ReleaseBuilder

The application is an executable JAR file that accepts the list of VCS revisions pointing to the source code elements to be included into an installation package or script. The main components of the application are the following:

- The data structure with source file types recognizable by ReleaseBuilder (e.g., SQL script, XML file, Oracle Forms, etc).
- The technology specific library of installation functions. It should provide installation functions for each type of the source file in a project, e.g., to install SQL file "INSTALLSQLFILE" function can be used.
- Interface for the generation of an installation package or script. Implementations of the interface are supposed to know a) the logic to determine the file type based on its name and location, b) the assignment of installation function, and c) the details of the generation of an installation script or package. For the purpose of the proof-of-concept implementation of the defined release building algorithm, the interface for the installation script generation is implemented for Oracle E-Business Suite projects.

Figure 3 outlines the major steps in ReleaseBuilder workflow. Repository connection details, the revision list from Analyzer and implementation of the interface for generation of installation package or script are provided as input parameters to ReleaseBuilder. As the first step, the tool performs the analysis of supplied revision list and prepares the list of affected files. Iterating over the file list the application identifies a necessary installation function and parameters and prepares an installation script or package.

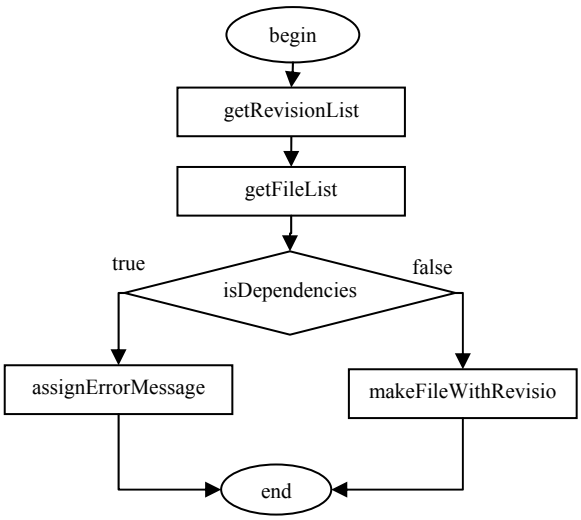

Fig. 2. The major steps of Analyzer

- getRevisionList – finds VCS revisions related to each issue
- getFileList – derives the modified and added files for each revision
- isDependencies – using VCS specific attribute it checks if the previous version of each file from the list is included into any of previous releases. If the previous revision of the file is not included into any of previous releases, the following information is persisted in a special data structure: filename, revision number, issue number
- makeFileWithRevisions – prepares the file with the list of VCS revisions

C. Limitations and Risks of the Method

The following limitations and potential risks related to the application of the proposed approach to building high quality releases from only tested changes should be taken into consideration.

1. The final installation package is produced from certain source code elements, leaving some changes not being included. However, during release testing the project quality assurance team usually works with configurations that include all changes. Moreover, the proposed method might produce the final installation package with untested configuration that was never installed to test. Thus, it is important to perform automatic unit and integration tests for final configuration in order to minimize the risk of broken deployment artifacts.
2. The mechanism used to determine relations of source code change is not reliable enough in current implementation of Analyzer tool. The situation, when the change in source code element "ComponentB" made to resolve issue "FeatureB" relies on the change in element "ComponentA" for issue "FeatureA", may lead to broken deployment artifacts, if "FeatureA" is excluded from release. The improvement of Analyzer tool is expected to minimize the aforementioned risks, but it might be reasonable to consider the additional check for this type of problem in a final code review depending on the project specifics and technologies.

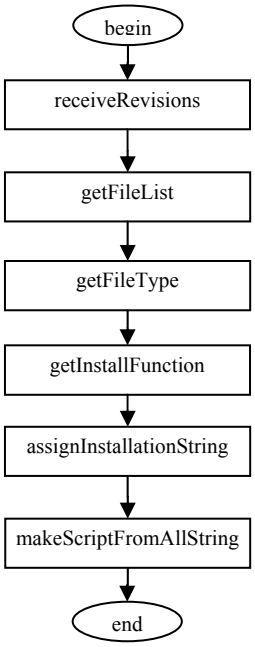

Fig. 3. The major steps of ReleaseBuilder

- receiveRevisions – receives VCS revision list as input
- getFileList – prepares the list of files related to each revision from the list
- getFileType – deduces the file type
- getInstallFunction – determines the installation function for the file
- assignInstallationString – prepares and saves the call of installation function for the file
- makeScriptFromAllString – completes the installation script generation routine that includes the call of installation functions for all files related to revisions supplied as an input parameter

D. Application of the Method in the Field of Insurance

Insurance Information Systems (IIS) should provide the opportunities for fast and easy modification and extension of its functions in order to support new products and changes in legislation. ISS supports the following main function of insurance business processes [4]:

- Insurance policy forming
- Claim proceedings
- Communication with re-insurance companies
- Data exchange between a central office and regional departments
- Insured risk monitoring
- Interconnection with accounting and other financial systems.

Taking into account that all these functions must be adapted to the requirements of selected insurance company and local conditions, the proposed algorithm can significantly improve the effectiveness of ISS release configuration. Currently the application of the proposed method is under testing within the framework of ERDF project (No.2011/0008/2DP/2.1.1.1.0/10/APIA/VIAA/018) "Insurance Distributed Software Development Based on Intelligent Agents, Modelling, and Web Technologies".

V. Related Studies

There are a few related studies that have influenced the proposed method for building high-quality releases. The current study completes the proposal described by us in [5], where the motivation and the basis of the approach to effective release configuration process aimed at supporting and ensuring effective maintenance of information technology products are discussed.

At the same time, this study presents some results of the ongoing joint research activities related to the improvement of systematic software processes that facilitate collaboration and allow reusing experience to address recurring problems successfully. The ideas discussed in the study are related and influenced by our work in [2] and [6] that describes the foundation of the framework for organizations that are moving towards a systematic reuse program and web-based information technology solutions.

VI. Conclusions

The paper discusses the method that aims to improve software configuration and release management processes for the maintenance projects, where the product is a large information system being maintained using one of the iterative development methodologies. The proposed concept relies on the possibility to prepare a release build that includes only tested changes. The release-building method proposed within the framework of this study receives a list of issues planned to be included in the release and produces the list of components to be included into the release build based on the information from the project code version control and issue tracking

systems. The provided proof-of-concept implementation of the method in Java can be easily adapted to any project that uses Subversion VCS, and the specifics of development process conforms to defined prerequisites.

The future research is related to a) improvements in the method implementation to work with other popular version control systems, b) verification and instructions for Analyser and ReleaseBuilder integration with popular issue tracking systems, and c) improvements in Analyser searching capabilities to allow deducing relations between different source files.

ACKNOWLEDGEMENT

This research is partly funded by the ERDF (ERAF) project (No.2011/0008/2DP/2.1.1.1.0/10/APIA/VIAA/018) "Insurance Distributed Software Development Based on Intelligent Agents, Modelling, and Web Technologies".

REFERENCES

[1] I. Sommerville, Software engineering. 6th. Addison-Wesley (2001)
[2] V. Kotov, A. Lesovskis and L. Novickis. Towards Reuse-Oriented and Web-Based Collaborative Framework for e-Business Providers. In: IFIP Advances in Information and Communication Technology, 2011, Volume 353/2011.
[3] Subversioning for Java: http://svnkit.com/. [Accessed September, 2012]
[4] M. Uhanova and L. Novickis, Application of Modelling and Internet Technologies in Marine Insurance Business. In: Proceedings of the International Conference COMPIT"2005, Hamburg, Germany, pp. 8-12, 2005.
[5] A. Bartusevics and V. Kotovs, Towards the Effective Reuse-Oriented Release Configuration Process. In: Proceedings of the 5th International Scientific Conference in Applied Information and Communication Technologies (AICT 2012). ISBN 978-9984-48-065-7), Jelgava, Latvia, , pp.99-104, 2012.
[6] V. Kotov, Reuse In Software Development Organizations In Latvia. In: Scientific Journal of RTU. 5. Series. Computer Science. vol. 43, pp. 90-96. RTU, Riga, 2010.

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

Vladimirs Kotovs is a doctoral student at Riga Technical University, the Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems. Vladimir Kotovs obtained his BSc (2004) and MSc (2007) degrees in Computer Science and Information Technology from Riga Technical University. His research areas are software reuse, software processes, information system configuration. He currently works at JSC Citadele Banka as a Java Developer.
Contact information: vladimir.kotov@gmail.com

Leonids Novickis is a Head of Division of Applied Systems Software. He obtained Dr.Sc.ing. degree in 1980 and Dr.Habil.Sc.ing. degree in 1990 from Latvian Academy of Sciences. He is the author of 180 publications. Since 1994 he has regularly been 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. He was an independent expert of IST and Research for SMEs in FP6 and FP7. He is a corresponding member of Latvian Academy of Sciences and an elected expert of Latvian Scientific Council. His research fields include Web-based applied software system development, business process modelling, e-learning and e-logistics.
Contact information: Leonids.Novickis@rtu.lv

Artūrs Bartusevičs, Vladimirs Kotovs, Leonīds Novickis. Metode efektīvai, uz atkārtotu lietošanu balstītai, programmatūras konfigurācijai un tās pielietojums apdrošināšanas jomā
Darbā ir atspoguļoti pētījuma rezultāti par sistemātisku programmatūras konfigurācijas un versiju pārvaldības procesu uzlabošanu. Tiek piedāvāts risinājums problēmai, kas var rasties, pielietojot iteratīvu metodoloģiju programmatūras projektu pārvaldībai, kad iterācijas beigās, kārtējā produkta versijas relīzes laikā, ne visi labojumi, kas ir veikti izejas kodā, ir notestēti. Tādā gadījumā var rasties nepieciešamība atlasīt tikai tos produkta komponentus, kuros visas veiktās izmaiņas ir notestētas esošās iterācijas gaitā. Darbā aprakstītā metode ļauj programmatūras uzturēšanas projektos veidot relīzes tikai no notestētiem labojumiem, pasargājot relīzes veidotāju no ilgstošām manuālām darbībām. Piedāvātā metode ir balstīta uz produkta izejas koda analīzi. Tās veiksmīgai pielietošanai ir nepieciešams, lai izstrādes organizācijā būtu ieviesta pieteikumu apstrādes sistēma, versiju kontroles sistēma un nepārtrauktās integrācijas process. Metode paredz produkta izejas koda elementu klasificēšanu atkarībā no izmaiņām, un nosaka, vai izmaiņas ir notestētas un var būt iekļautas gala produktā. Nākamajā solī no identificētiem elementiem tiek būvēts produkts un instalēts testa instancē. Veiksmes gadījumā tiek izpildīti automātiski integrācijas un vienību testi, lai pārliecinātos, ka savāktās konfigurācijas atbilst pieņemtajiem kvalitātes kritērijiem. Darbā tiek analizēti arī riski, kas saistīti ar metodes ieviešanu praktiskā izmantošanā un aprakstīti to mazināšanas pasākumi. Pētījuma ietvaros ar JAVA programmēšanas valodas palīdzību ir izstrādāti rīki, kuri tiek izmantoti ar relīzes veidošanas metodi darbam ar versiju kontroles sistēmu Subversion. Attīstot turpmāk šo metodi, ir plānota rīku uzlabošana, to iespējamai izmantošanai arī ar citām versiju kontroles sistēmām, kā arī metodes pilnveidošana atkarību meklēšanas kontekstā.

Артур Бартусевич, Владимир Котов, Леонид Новицкий. Метод для эффективной конфигурации релиза программного обеспечение и его применение в области страхования
В работе представлен результат исследования, связанного с систематическим улучшением процессов конфигурации программного обеспечения и управления выпуском релизов программных продуктов. Предлагается решение проблемы, возникающей при использовании итеративной методологии для управления проектами, если на момент запланированного выпуска продукта не протестированы все исправления, которые были включены в исходный код. В таком случае возможно внедрение новой версии с использованием только полностью протестированных компонентов продукта. Предложенный метод дает возможность повысить качество внедряемых релизов для проектов по сопровождению программного обеспечения, а также снижает необходимость в рутинной ручной работе по созданию инсталляций. Представленный метод основан на анализе исходного кода продукта. Для его успешного применения в организации необходимо использование системы регистрации дефектов, системы управления версиями и непрерывный процесс интеграции программного обеспечения. Предлагаемый подход подразумевает классификацию элементов исходного кода, в зависимости от внесенных в них изменений, а также определение того, были ли протестированы изменения и возможно ли их включение в конечный продукт. На следующем этапе происходит создание инсталляции из идентифицированных элементов и установка в среду для тестирования, где выполняется набор интеграционных и юнит тестов для проверки критериев качества собранной конфигурация. В статье также анализируются возможные риски, связанные с практическим применением метода, и описываются меры по их предотвращению. В рамках исследования созданы несколько инструментов на языке Java для использования предлагаемого метода с системой контроля версий Subversion. Дальнейшее возможное развитие метода и инструментов связано с улучшением их использования с другими системами контроля версий.