Towards Requirements Variability in Agile Software Product Line Development

Oleh Tovstokorenko[1], Rustam Gamzayev[1]

[1] National Technical University "Kharkiv Polytechnic Institute",
Kyrpychova str., 2, Kharkiv, 61002 Ukraine
{tovstokorenko@gmail.com, rustam.gamzayev@gmail.com}

Abstract. This article proposes an approach to support an agile development of software product lines (SPL) using requirements variability management within a Scrum methodology. The main aim is to classify all requirements in 3 types according to the typical SPL-components: core, variable and new ones, and in this way to reduce a sprint backlog for any Scrum iteration. An information base for this approach is structured, its role in common Scrum method is shown, and a conceptual scheme for requirements variability management is proposed.

Keywords: SPL, agile, requirement, variability, software, traceability, Scrum

Key terms: requirements variability, software product lines, traceability

1 Introduction: Problem Actuality and Research Aims

Modern software development systems are built with a number of services, components and classes that could be reused. This is especially important during development of the similar systems and requires establishing the requirements similarity identification mechanism. Often such problems are poorly formalized and require a program-analytical solution approach. The obtained solutions lead to a certain degree of design structures similarity and it becomes possible to say that differences arise due to variations in software artifacts. During SPL development, variability management process is located between requirements elicitation and actual components implementation[1]. In particular, during implementation of some projects in SPL, it is necessary to determine similar requirements. As a result the initial requirements catalog (RC) can be compiled. Having first versions of RC it becomes possible to start development stage. All SPL components can be divided into 3 groups [2] (Fig. 1).

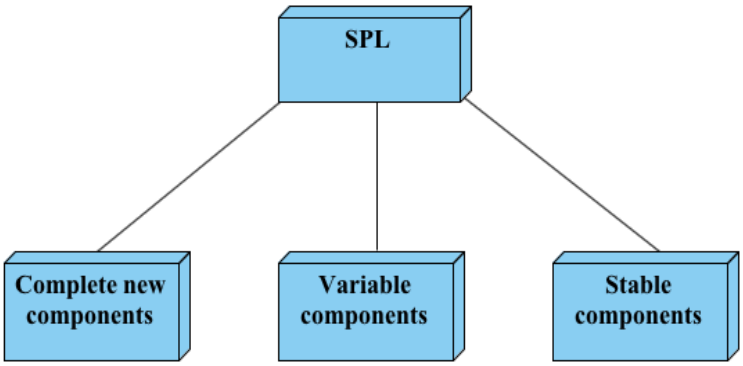

Fig. 1. Components categories for products within the SPL framework

Each group differs based on an amount of reused code that has to be modified in order to implement new requirement. If artifacts similarity value less that 25% - it can be added to the group of new components, more than 25% - variable components and without changes – reusable components [2]. It is clear that the value of similarity between requirements and implemented components (by these requirements) can be different. Interdependence between these values allows to apply a variability management approach during requirements analysis stage.

In particular, in [3] it is shown that due to presence of links between requirements and artifacts, defined with advanced traceability matrix (ATM), it was possible to improve the completeness and accuracy of requirements traceability in agile software development by developing knowledge-oriented models and information accumulation technology, analysis and management of data about user requirements. However, the proposed in the [3] approach was applied for the single product development activity and has no possibility to apply data between different projects. Thus, the goal of this research is to elaborate a process for tracing requirements during the SPL development to investigate a possible connection between value of similarities between requirements and similarities between components implemented by current requirements.

2 Related work

The comparisons of Agile and SPL methods was done in [4], where it was sum up that both methods tailored to produce high quality software solutions, however there are a number of differences and it was proposed to use Agile Software Product Line Method. Variability management is considered as one of the main objectives in the development of software product lines. It is gained its most extensive grows during the SPL developing process in [1] reported about new advantages of adoption of Model Driven Engineering (MDE) paradigm in variability specification process. The global picture is a sequence of models from requirements to features, and from both of these to architecture (a UML model). In [5] presented that a self-contained model to explicitly represent the product-line variability in one central model, and further

enables the definition of variability in different requirements models. In [6] the authors describe techniques for generating requirements specifications with variability from so-called requirements library. The research results described originate from a process improvement initiative at DaimlerChrysler. The presented approaches are therefore pragmatic and aimed at current industrial practice, but they are formally based on a category-theoretical notation.
From these papers and other sources that we have analyzed it becomes clear that SPL development and Agile-methodology could be used together, but there is no precisely define procedure for requirements variability in the Agile SPL development.

3 The Proposed Variability Requirement Management Framework for Agile SPL Development

3.1 Specific Features of SPL and their Impact on Requirements Variability Management

SPL development usually consist of 2 phases. The first phase is responsible for a domain engineering [7] and the second one provides application engineering. Domain engineering plays one of the main roles to define different types of domain elements and to assign it either to core elements that stay unchangeable or contain some degree of commonality and assigned to group of variable elements. Even when variation points are defined, it could be not straight – forward task to extract source code that is required for variable requirement. In order to perform such operations, we should think about building software architectures suitable for variability managements, using some known design paradigms like aspect-oriented development [8] or some development methodologies like domain-driven development [9]. To make possible a traceability between requirements and implemented components, it is necessary to take into account additional information, which can be obtained both directly during programmer's work, by adding additional software solutions, and conducting additional data analysis and identifying new knowledge. For example, we can get information about the links of variability model elements, with sets of requirements and artifacts. Such kind of information is possible to use as information basis for the traceability model. Thus, to provide variability management for requirements management stage in an effective and correct way the appropriate operating model (OM) can be proposed [10]. With the help of OM supposes to build traceability model regarding to connect SPL requirements and implemented components. This OM uses the appropriate methods and metrics which can be presented as a tuple:

$$OM(RVM) = < InfBase, Algorithms, Metrics > \quad (1)$$

$OM(RVM)$ – Operational model for requirements variability management in Agile SPL.

$InfBase$ – Information basis consists of requirements, software artifacts, information about project iterations and software developers roles.

$Algorithms$ – commonality analysis algorithms for software requirements and requirements classification algorithms.

$Metrics$ – a set of metrics like Defect count, Technical Debt, Running Automated Tests, etc [11] to estimate quality of the agile process.

Thus, proposed OM should allow to determine formal process of requirements variability management in Agile SPL, that will open a possibility to collect and analyze information for SPL variability management, based on data from *InfBase*, using commonality algorithms and *Metrics*.

3.2 Information Basis for the Proposed Framework

In the paper [12] it was shown, that information basis for the traceability model could be represented like on (1). In this formula, InfBase is a data set, that contains detailed description of artifacts from each SPL development stages, such as requirements, software artifacts (e.g. product components).

Then, using commonality analysis algorithms, based on data from InfBase should become possible to distinguish and classify sets of requirements in connection to implemented components. The result of this extracting and classification with the Metrics data will become an input data for requirements variability management in Agile SPL.

$$InfBase = \langle D, R, F, S, W \rangle, \qquad (2)$$

where $D = \{d_i\}, i = \overline{1,I},\ I =\ |\mathrm{D}|$ is a set of software developers;

$R = \{d_j\}, j = \overline{1,J},\ J =\ |\mathrm{R}|$ is a set of requirements;

$F = \{f_k\}, k = \overline{1,K},\ K =\ |\mathrm{K}|$ is a set of software artifacts,

$S = \{S_l\}, l = \overline{1,L},\ L =\ |\mathrm{S}|$ is a set of project sessions (iterations);

$W = \{w_m\}, m = \overline{1,M},\ M =\ |\mathrm{W}|$ is a set of working profiles or developers roles.

This information is used only for a single legacy project developed, in order to apply this approach for SPL development and variability in different components then additional information should be considered. In this case, the information basis given in (1) has to be extended and can be represented in the following way:

$$InfBase_P^* = \langle D, R, F, S, W, C \rangle, \quad (3)$$

where $C = \{C_z\},\ z = \overline{1,Z},\ Z = |C|$ is a set of commits into version control system (SVN, GIT or Mercurial) connected with implemented requirement and software artifacts changed during commit; Additional element C in the expression (3) is used to define relationship between software artifacts implemented in different projects of the SPL development and target requirement that could be represented as user story. We are assuming that artifacts of the commit could be represented as a single feature that could be reused in the future. Each requirement has relationship to one or many commits and each commit has relationship to the one or more artifacts that were changed.

$P = \{P_u\}, u = \overline{1, U},\ U =$ |P| is set of Projects in the SPL.
$InfBase_P^*$ should be build for each Project P, and used to analyze constant, variable and similar components. Thus, by extending existing model, it becomes possible to extract additional information that will be used to identify reusable artifacts or elements of source code. The next part presents a conceptual diagram of the proposed process.

3.3 Conceptual Scheme for Agile Requirements Variability in Scrum Methodology

Various agile software methodologies used to organize development and one of the most commonly used is Scrum [13]. It is organized in an iterative and incremental way; there is a Product backlog (PB) that contains initial requirements and Sprint backlog (SPB) that contains validated, verified and prioritized requirements.

Scrum does not define the content criteria of requirements. This process assumes the presence of a product backlog and sprint backlog. The product backlog consists of users stories that most often represented in text form. Traditional Scrum cycle could be extended taken into account a processes proposed in Section 3.2 and used in a context of SPL development (Fig. 2).

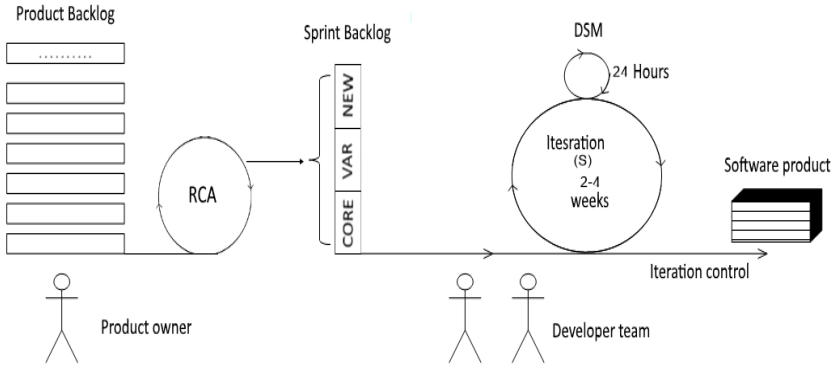

Fig. 2. The requirements variability scheme within agile development of SPL

According to the presented Agile SPL schema, new RCA process is responsible for assessing the degree of requirements similarity that require either PB with represented with user stories in the textual way or it could be provided additional step after PB is prepared that will transform requirements to some model using UML or FODA notation that should simplify RCA.

4 Classification Approach for Requirements Variability Management

New process is called Requirements Classification Approach (RCA). This process represents an algorithm in OM (1). It used to analyze input requirements from PB and classify them to three predefined groups: “Core”, “Variable” and “New” described in the Fig. 3.

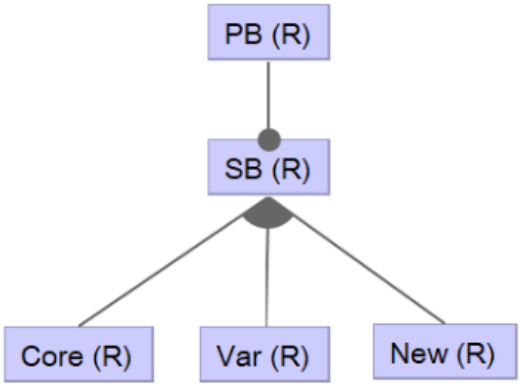

Fig. 3. Requirements Classification Approach

The proposed FODA model represents a hierarchy of properties (features) in the domain “*Agile SPL Requirements Variability*” using the domain concepts and sub-concepts, which are represented with specific notation elements given in [14], namely:

1) at the root of this hierarchy the main conceptual feature PB (R) is shown: this is a PB including a set of initial requirements R;
2) any PB (R) according to Scrum method must be represented as a sequence of SB(R) at the 2nd hierarchy level, that is why this is a mandatory feature marked with black bullet, and the appropriate multiplicity is given with 1…*, it means: at least one SB(R) must be constructed from PB(R) in order to start a sprint session within any Scrum project;
3) at the 3rd level the terminal nodes represent 3 possible alternative sub-concepts included in any SB(R), they can be accordingly:
    - a mandatory feature Core (R). it means: any SB (R) includes at least 1 core requirement to be implemented in SPL;
    - one or two optional features: Var (R) and / or New (R) both marked with empty bullets.

Presented separation depends on amount of changes for each requirement, relatively to already implemented requirements during SPL development [15]. For now, classification rules are not defined in details. But in [16] it was shown that

source code reusability metric could be calculated in order to perform this step. In more details the process of classification described in the Fig. 4.

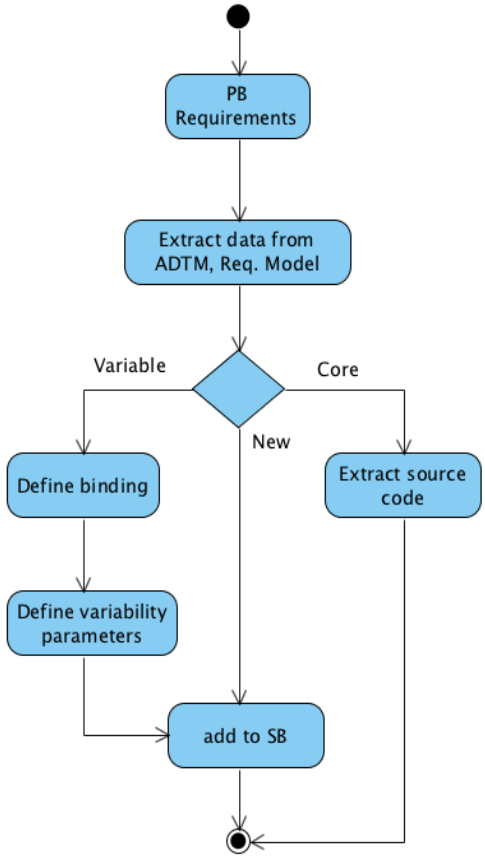

Fig. 4. Requirements Classification Approach

We execute RCA for each requirement when it is extracted from PB. First, we need to check if this requirement was implemented before (need to check in the $InfBase_P^*$). If commonality will found, then this requirement can be assigned to "Core" group and reused later. In other case, we need to check if it is a new one and add it to the SB. In other case it belongs to the the "Variable" group and appropriate requirements will be extracted from repository. It should be adapted with respect to previous implementations.

Identifying requirements similarity can be done in several ways, most widely used one is based on FODA notation. When requirement description similarity value will be known, the result data will become a basis data for more complex analysis, that can be done by analytics. The result of described RCA process is the Requirements Variability Model, that will additionally include data about implemented source artifacts by each requirement and data about quality of result implementation. It can be done semiautomatic cause it based on source code analysis that can be automated.

Thus, proposed framework should provide opportunity to accumulate information about implemented requirements, as a result, it will be possible to reuse components on the requirements analysis stage.

5 Conclusions and Future Work

We have presented the approach to support a requirements variability management which aims to reduce a sprint backlog size in agile SPL development. In future dissertation research requirements classification process will be considered in more details. This will require to inspect different notation of requirements specification and metrics for commonality and similarity analysis. Additionally, it is necessary to analyze how artifacts connected with given requirements are mapped to the feature and what architecture, package and class structure required to successfully build SPL. In addition, the possibilities to extend current process by quality analysis will be observed. The results of quality analysis are going to become an additional parameter for classification approach. Implementation of the proposed process should allow to collect solutions with a certain quality level and to be reused basing on the information retrieved during software development in SPL. We are planning to implement a software prototype to improve the SPL development process based on the proposed approach.

6 References

1. Miguel A. Laguna, Bruno Gonzalez-Baixauli: Requirements variability models: meta-model based transformations. Proceedings of the 2005 symposia on Metainformatics, Esbjerg, Denmark (2005)
2. Martinkus, I. Information technology for the development of software product lines based on methods and tools of domain modeling. - Manuscript. Dissertation for the candidate's degree on the speciality 05.13.06 – information technologies. – National Technical University «Kharkiv Polytechnic Institute», Kharkiv, 2018.
3. Tkachuk M. V., Gamzayev, R.O., Mayr H.C. et al. Models and Tools for Effectiveness Increasing of Requirements Traceability in Agile Software Development // Problems of Programming. – Kiyv: Ukrainian Academy of Science. – 2012. – No. 2–3 (special issue). – p. 252 – 260.
4. Tian, K. and K. Cooper, Agile and Software Product Line Methods: Are They So Different?, in 1st International Workshop on Agile Product Line Engineering (APLE'06). 2006, IEEE Computer Society: Baltimore, Maryland, USA.
5. Bühne S, Lauenroth K, Pohl K (2004) Why is it not sufficient to model requirements variability with feature models. In Proceedings of the workshop: automotive requirements engineering (AURE04)
6. Tavakoli Kolagari R., Reiser MO. (2007) Reusing Requirements: The Need for Extended Variability Models. In: Arbab F., Sirjani M. (eds) International Symposium on Fundamentals of Software Engineering. FSEN 2007. Lecture Notes in Computer Science, vol 4767. Springer, Berlin, Heidelberg
7. Bosch J., Florijn G., Greefhorst D., Kuusela J., Obbink J.H., Pohl K. (2002) Variability Issues in Software Product Lines. In: van der Linden F. (eds) Software Product-Family

Engineering. PFE 2001. Lecture Notes in Computer Science, vol 2290. Springer, Berlin, Heidelberg
8. Filman, R. and D. Friedman. "Aspect-oriented programming is quantification and Obliviousness." Proceedings of the Workshop on Advanced Separation of Concerns, in conjunction with OOPSLA'00 (2000)
9. Evans, Eric (2004). Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley. ISBN 978-032-112521-7. Retrieved August 12, 2012.
10. De Vries, M. A Method for Identifying Process Reuse Opportunities to Enhance the Operating Model / M. de Vries, A. van der Merve, P. Kotze, A. Gerber // IEEE International Conference on Industrial Engineering and Engineering Management, 2011.
11. Johan Gustaffson, Chalmers University of Technology. University of Gothenburg. Department of Computer Science and Engineering. Göteborg, Sweden, June 2011. Model of Agile Software Measurement: A Case Study. Master of Science Thesis in the Programme Software engineering and. Technology.
12. Gamzayev R.A. Models and information technology for requirements traceability in agile-software development. - Manuscript. Dissertation for the candidate's degree on the speciality 05.13.06 – information technologies. – National Technical University «Kharkiv Polytechnic Institute», Kharkiv, 2013.
13. Ambler, S.W.: The Agile System Development Lifecycle (SDLC) (2005). Accessed on April 9, 2018, http://www.ambysoft.com/essays/agileLifecycle.html
14. Riebisch M., "Towards a more precise definition of feature models", Modelling Variability for Object-Oriented Product Lines, , 2003, p. 64–76.
15. Paliwal, N., Shrivastava, V.: An approach to find reusability of software using objet oriented metrics. International Journal of Innovative Research in Science, Engineering and Technology 3 (2014) 8. Pohl, K., B¨ockle, G., Linden, F.: Software product line engineering: Foundations, principles, and techniques. Springer p. 467 (2005)
16. Tkachuk, M., Martinkus, I., Gamzayev, R., Tkachuk A. An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development / ed. Heinrich C. Mayr, Martin Pinzger. INFORMATIK 2016. Lecture Notes in Informatics (LNI). Vol. P-259: Kollen Druck+Verlag GmbH, Bonn, 2016. pp. 143-156.