ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/341113798

SELECTION OF OPTIMAL SOFTWARE DEVELOPMENT METHODOLOGY BASED ON WEIGHTED AGGREGATED SUM PRODUCT ASSESSMENT METHOD

Chapter · December 2019

CITATIONS
0

READS
5

2 authors:

Darjan Karabasevic
University Business Academy in Novi Sad
94 PUBLICATIONS **700** CITATIONS
SEE PROFILE

Florentin Smarandache
University of New Mexico Gallup
3,119 PUBLICATIONS **23,775** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Types of fuzzy graphs and their applications in decision making View project

Project Neutrosophic sets View project

All content following this page was uploaded by Darjan Karabasevic on 03 May 2020.

The user has requested enhancement of the downloaded file.

SELECTION OF OPTIMAL SOFTWARE DEVELOPMENT METHODOLOGY BASED ON WEIGHTED AGGREGATED SUM PRODUCT ASSESSMENT METHOD

***Darjan Karabašević*[1], *Florentin Smarandache*[2]**

[1]Faculty of Applied Management, Economics and Finance,
Business Academy University, Jevrejska 24, Belgrade, Serbia,
darjan.karabasevic@mef.edu.rs
[2] Department of Mathematics, University of New Mexico, 705
Gurley Avenue, Gallup, NM 87301, USA,
fsmarandache@gmail.com

Abstract: The software development methodology covers the complete software life cycle. It involves the production of quality and reliable software in a systematic, controlled and efficient manner using formal methods for specification, evaluation, analysis and design, implementation, testing, and maintenance. Today, software is used in all domains of education. From primary and secondary schools to higher education institutions are using specialist software packages intended for research. The aim of this manuscript is a selection of the software development methodology based on multiple-criteria decision-making methods. PIPRECIA method is applied for defining the weights of the criteria, whereas WASPAS method is applied for the ranking of alternatives. The application of the proposed approach, as well as its efficiency and effectiveness, are shown in the conducted case study.

Keywords: Software, Software development methodology, MCDM, WASPAS
JEL: D81, C44

1. INTRODUCTION

The software development methodology covers the complete software life cycle. It involves the production of quality and reliable software in a systematic, controlled and efficient manner using formal methods for specification, evaluation, analysis and design, implementation, testing, and maintenance. Today, software is used in all domains of education. From primary and secondary schools to higher education institutions are using specialist software packages intended for research.

Manger (2012) states that software product is a collection of computer programs and related documentation, created precisely for the reason of being sold. It can be developed for a specific user (customized product) or generally for the market (generic product). Today's software includes the necessity that he must be of good quality. More specifically, a software product is expected to be characterized by the following quality attributes: a) Maintenance; b) Reliability and security; c) Efficiency; d) Usability.

Thus, the software represents a series of commands that are stored in the computer's memory. It is executed on hardware and is required for proper operation and functioning of a computer system. According to the purpose one of the most common software division is into two groups: a) System software: all programs, software packages, etc. intended for the functioning of a computer system are belonging to system software; b) Application software: all programs, software packages, etc. intended to solve specific problems and tasks of computer system users are belonging to application software (Tomašević, 2012).

The pace of change in the software development industry is still high. People continue to push the boundaries of known techniques and practices to develop as efficient and effective software as possible. Software development lifecycle models and business decision models contribute to controlling product development in different ways.

Software products are among the most complex systems made by man. Therefore their development requires the use of techniques and processes that can be successfully scaled up to very large applications while satisfying demands for size, performance and security, all within the time and budget constraints. The complexity of large software systems is overcome by the use of higher-level abstraction structures, such as software architecture.

Thus, a set of activities that are related to the "initialization, design, realization and sale of software products and managing all the resources that are related to that product" is called software engineering which is of crucial importance to software development (Steward, 1987).

Today, decisions are made daily and are one of the most important elements of management activities. With globalization and increasing business dynamics, changes have been made in the decision-making process, so decision-making has become much more demanding and complex. Multiple-Criteria Decision-Making (MCDM) represents the process of selecting one alternative from a set of available alternatives or, in some cases, ranking alternatives based on a predefined set of specific criteria that most often have different significance. Justification for the application of the MCDM methods is relevant approach to making decisions and the adoption of sustainable solutions (Stanujkić *et al*., 2019a; Stanujkic *et al*., 2019b; Karabašević & Maksimović, 2018; Stanujkić *et al*., 2017b).

Therefore, the main aim of this manuscript is to present an approach for the selection of software development methodology based on MCDM methods. For the defining weights of the criteria, PIPRECIA method is applied, whereas for the purpose of the ranking of alternatives WASPAS method is applied.

2. THEORETICAL BACKGROUND

Nowadays, software engineering is receiving increasing attention in software development. Software engineering is a systematic approach to the development, exploitation, maintenance, and replacement of software products. Software engineering is a technological and management discipline that deals with the systematic production and maintenance of software products, which should be developed on time and at an estimated cost (Mead, 2009).

Characteristics of software engineering are (Buckley, 1987): a) modeling as the basis of design (object-connection model, process model, data model); b) methods of analysis, synthesis, and identification are used; c) division into levels, related to the phases of the software product life cycle and the project phase; d) interdisciplinarity and user involvement; d) project organization; e) validation and verification of results, quality assurance, presentation of results, documentation.

Because of all of the above, software engineering requires the use of both analytical and descriptive tools that have been developed within the computer sciences, along with the rigorous approach that engineering disciplines bring to achieving adequate reliability and security, all through the teamwork of software engineers working in a cohesive environment. At today's level of software engineering development, an organization's ability to handle software development in this way is precisely measurable and ranges from level 1 where software processes are unpredictable to level 5, where software processes are optimized.

The software development model is selected depending on the nature of the project and the application, the technical orientation of the people who will participate in the development, the methods and tools that will be used in the development, the methods of control as well as the products that are required. The primary goal of model creation is to provide software products that meet user requirements.

Depending on the importance of particular stages and activities of software development and the forms of organization and development management, as well as the experience of employees and the nature of the product, there are (Balaji & Murugaiyan, 2012; Martin, 2002):

a) Sequential software development model, the so-called *waterfall*;

b) Iterative and incremental model of software development;
c) Agile development model.

The purpose of the SWEBOK project is to characterize the content of software engineering as a discipline, as well as to differentiate software engineering from other disciplines such as computer science, project management, computer engineering, etc. According to the SWEBOK project, software engineering methods can be divided into three areas: a) heuristic methods relating to methods based on the informal approach; b) formal methods based on a mathematical approach; and c) prototyping methods relating to methods based on different types of prototyping (Antović *et al*., 2008; Stanojević *et al*., 2006).

Each model of the software development process uses a requirement specification as input and a delivered product as an output. Over the years, many such models have been proposed. Below, some of the most popular will be discussed in order to better understand their similarities and differences.

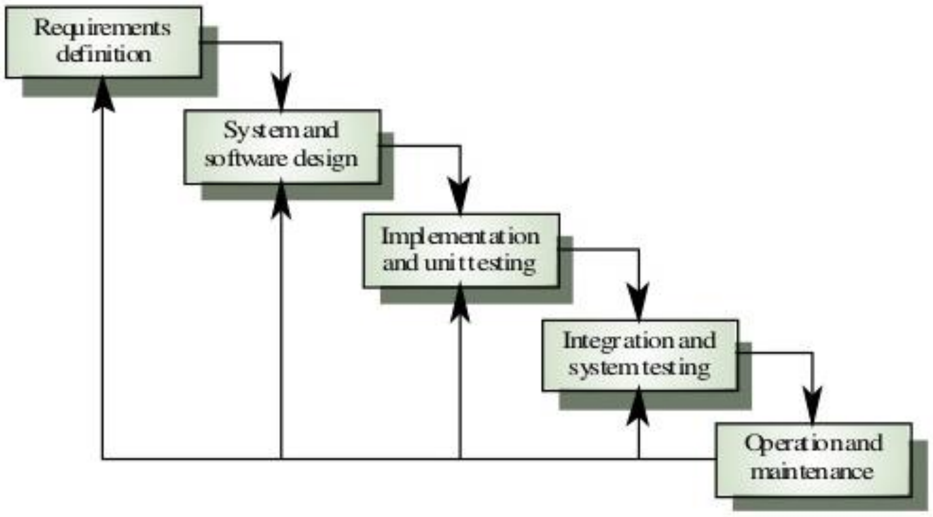
2.1. The waterfall model

One of the first proposed is a waterfall model. The waterfall model implies that it is necessary that one phase of the development must be fully completed before the beginning of the next one (Royce, 1970).

The waterfall model is very useful as an aid in expressing what the software development team needs to do. Its simplicity makes it easy to provide explanations to those unfamiliar with the software development process as it explicitly indicates among the steps necessary to begin the next phase. Many other more complex models represent a "beautified" waterfall model, through the inclusion of feedbacks and activities (Balaji & Murugaiyan, 2012). The biggest problem with the waterfall model is that it does not reflect the actual way in which the code evolves. Except perhaps for the very clear problems, the software is usually developed through a number of iterations. Software is often used to solve a problem that has never been resolved before, or whose solution must be improved to reflect changes that have occurred in the business or work environment (Pfleeger & Atlee, 1998).

Although this model has been used for many years in the production of many quality systems, it does not mean that no problems occur. In recent years, the model has been criticized for its rigidity and inflexible procedure.

Figure 1. *Waterfall model*


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

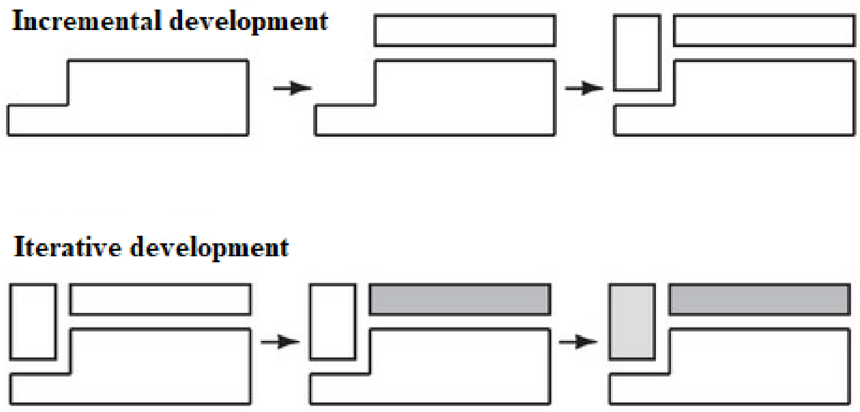
Source: https://www.slideshare.net/Ehtesham17/waterfall-model-in-software-engineering

2.2. The iterative and incremental model

The problems encountered in the waterfall model led to the need for a new method of system development that would provide quick results, which would require less initial information and offer more flexibility.

When iterative development is applied, the project is divided into smaller parts. This allows the development team to demonstrate results early in the process and to receive valuable feedback from system users. Often, each iteration is actually one mini-waterfall with feedback from one phase that provides vital information for the next phase. In iterative development, iteration is delivered immediately, at the very beginning, and then the functions of each subsystem are changed, in each new version. In incremental development, the system as specified in the requirement specification is subdivided into subsystems by functions. Versions are defined initially as small, functional subsystems, and then new features are added to each new version (Larman, & Basili, 2003).

Figure 2. *Incremental and iterative development*



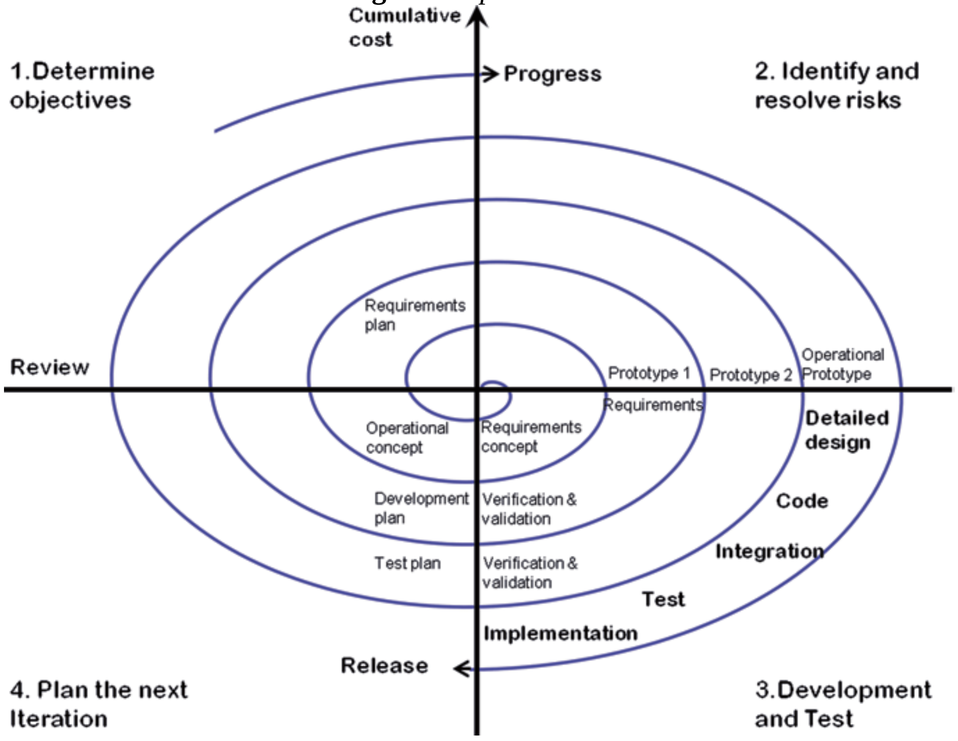
Source: http://www.link-university.com/lekcija/Fazni-razvoj-i-spiralni-model/2620

2.3. The spiral model

Boehm (1988) observed the development of software in light of the risks involved, suggesting that the spiral model can combine development activities with risk management, in order to be smaller and easier to control. This model is designed to include the best features of the waterfall model and introduces a new component - the risk assessment. The term "spiral" is used to describe the process that follows the development of the system.

The spiral model, shown in Figure 3, in some ways is similar to the iterative development shown in Figure 2. Starting with the requirements and initial development plan (including budget, constraints and alternatives in terms of staff, design and development environment), this process introduces a risk assessment step and a prototype alternatives, before producing a "working principles" document, to describe the functioning of the system at a high level of abstraction. From that document, a set of requests is defined and monitored to verify the completeness and consistency of the request. Therefore, the principle of operation is the product of the first iteration, while the requirements are the main product of the second iteration. In the third iteration, system development produces the design, while the fourth iteration enables testing (Fliger *et al*., 2006).

Figure 3. *Spiral model*



Source: Cowley (2014)

In each iteration, the risk analysis identifies different variants in terms of requirements and constraints, while prototyping verifies the feasibility or desirability of selecting the appropriate variant. After identifying the risks, project managers must decide how to eliminate or minimize them. To avoid the risk of choosing interfaces that would prevent productive use of the new system, designers can prototype both interfaces and test them in order to determine which is preferable, or even include both interfaces in the project so users can, after logging in, select interface. Restrictions such as budget and delivery time help in choosing a risk management strategy (Neill & Laplante, 2003).

2.4. The agile methods

Agile methodologies emerged in the late 90s when a group of software engineers concluded that previous approaches and methodologies for software development were not suitable in a turbulent environment and that it was not possible to bind and achieve firm delivery times for software solutions and customer satisfaction. They met and through mutual exchange of opinions came to the basic principles of agile methodologies, which they wrote down in the so-called Agile Manifesto (Jovanović *et al*., 2016).

Kilibarda *et al*. (2016) find that agile methodologies differ from traditional ones in that they require the development of software products through shorter development cycles. With its completion, it is possible to deliver one piece of a software product to the client and make the necessary changes with it, in order to reach the final result faster and more efficiently.

In the practice of implementing agile methodologies for software product development, a number of methods are proposed and used today. The most famous are Scrum, Extreme Programming (XP), Crystal Clear, DSDM, and more. The Scrum method is one of the most popular and in practice the most used method of agile software development management.

2.4.1 SCRUM methodology

This method is based on the basic principles that characterize the agile approach and is convenient in practice because it is very easy to use. This method suggests that software development work takes place in shorter cycles called sprints, followed by ongoing consultations with the client, and that, after a certain cycle, analysis and review will be carried out and, if necessary, the desired and necessary changes will be made. This includes mandatory meetings before and after each sprint, in order to consider whether everything was done accordingly to the requirements and if it is necessary to introduce some changes. In a particular situation, it is possible to go back and implement a specific sprint according to new requirements (Jovanović *et al*., 2016; Pichler, 2010).

Development cycles - sprints are time intervals that can last one month, usually lasting two or more weeks. The software development team using the SCRUM method has special authority in terms of organizing and operating, as well as the special member or product owner that has certain authorization and responsibilities regarding the work of the development team and delivering the desired results to the client. In addition to team members working on software development, the SCRUM methodology envisages two specific roles related to team operations. These are the product owner and SCRUM master (moderator or mediator) (Jovanović *et al*., 2016).

One of the most commonly used and researched methods is SCRUM, which describes an iterative development process with the gradual delivery of value. SCRUM methodology can only reach its full potential if all elements are well defined with fully dedicated teams.

3. METHODOLOGY

Weighted aggregates sum product assessment (WASPAS) method was developed by Zavadskas *et al*. (2012). The WASPAS method represents a unique combination of two MCDM approaches weighted sum (WS) method and weighted product (WP) method.

In order to cope with a wider range of problems, the WASPAS method has many extensions, such as: WASPAS-G (Zavadskas *et al*., 2015), WASPAS-IFIV (Zavadskas *et al*., 2014), WASPAS-SVNS (Baušys & Juodagalvienė, 2017), WASPAS-IFN (Stanujkic & Karabasevic, 2018); WASPAS-R (Stojic *et al*., 2018).

Also, until now WASPAS is applied for solving the most diverse problems, such as: manufacturing decision making (Chakraborty *et al*., 2014), construction site selection (Turskis *et al*., 2015), personnel selection (Karabasevic *et al*., 2016; Urosevic *et al*., 2016), website selection (Stanujkic & Karabasevic, 2018), and so on.

The computational procedure of WASPAS method can be precisely presented as follows (Karabasevic *et al*., 2016; Urosevic *et al*., 2016):

Step 1. Determine the optimal performance rating for each criterion. In this step, the optimal performance ratings are calculated as follows:

$$x_{0j} = \begin{cases} \max\limits_i x_{ij}; & j \in \Omega_{\max} \\ \min\limits_i x_{ij}; & j \in \Omega_{\min} \end{cases}, \tag{1}$$

where x_{0j} denotes the optimal performance rating of *j*-th criterion, $\Omega_{\max}$ denotes the benefit criteria, i.e. the higher the values are, the better it is; and $\Omega_{\min}$ denotes the set of cost criteria, i.e. the lower the values are, the better it is, *m* denotes number of alternatives; *i* = 0, 1, ..., *m*; and *n* denotes number of criteria, , *j* = 0, 1, ..., *n*.

Step 2. Construct the normalized decision matrix. The normalized performance ratings are calculated as follows:

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{x_{0j}}; & j \in \Omega_{\max} \\ \dfrac{x_{0j}}{x_{ij}}; & j \in \Omega_{\min} \end{cases}, \tag{2}$$

where r_{ij} denotes the normalized performance rating of *i*-th alternative in relation to the *j*-th criterion.

Step 3. Calculate the relative importance of i-th alternative, based on WS method. The relative importance of *i*-th alternative, based on WS method, is calculated as follows:

$$Q_i^{(1)} = \sum_{j=1}^{n} w_j \, r_{ij} \, , \tag{3}$$

where $Q_i^{(1)}$ denotes the relative importance of *i*-th alternative in relation to the *j*-th criterion, based on WS method.

Step 4. Calculate the relative importance of i-th alternative, based on WP method (Madić, 2014). The relative importance of *i*-th alternative, based on WP method, is calculated as follows:

$$Q_i^{(2)} = \prod_{j=1}^{n} r_{ij}^{\ w_j} \tag{4}$$

where $Q_i^{(2)}$ denotes the relative importance of *i*-th alternative in relation to the *j*-th criterion, , based on WP method.

Step 5. Calculate total relative importance, for each alternative. The total relative importance, or more precisely the joint generalized criterion of weighted aggregation of additive and multiplicative methods is calculated as follows:

$$Q_i = 0.5Q_i^{(1)} + 0.5Q_i^{(2)} = 0.5\sum_{j=1}^{n} w_j \, r_{ij} + 0.5\prod_{j=1}^{n} r_{ij}^{\ w_j} \tag{5}$$

In order to have increased ranking accuracy and effectiveness of the decision making process, in WASPAS method, a more generalized equation for determining the total relative importance of *i*-th alternative is developed as below:

$$Q_i = \lambda Q_i^{(1)} + (1-\lambda)Q_i^{(2)} = \lambda\sum_{j=1}^{n} w_j \, r_{ij} + (1-\lambda)\prod_{j=1}^{n} r_{ij}^{\ w_j} \tag{6}$$

4. CASE STUDY OF THE SELECTION OF SOFTWARE DEVELOPMENT METHODOLOGY

This section will present a case study of the selection of software development methodology based on the use of PIPRECIA and WASPAS methods.

Based on the literature review, alternatives that will be evaluated are as follows: Waterfall methodology – A_1; Iterative and incremental methodology – A_2; Spiral methodology – A_3; and SCRUM methodology – A_4.

Pivot Pairwise Relative Criteria Importance Assessment method (PIPRECIA) method is developed by Stanujkic *et al.* (2017a) and is used for the determination of the weights of the criteria. Based on the research carried by Mahapatra and Goswami (2015), in this manuscript following criteria were determined, namely: Requirement analysis – C_1; Status of the development team – C_2; User's participation – C_3; and Project type and associated risk – C_4.

Step 1. Determination of weights of criteria

Responses and assigned weights of the evaluated criteria obtained from the three Decision Makers (DMs) by applying PIPRECIA method are shown in Table 1-3, whereas in Table 4 are shown group weights.

Table 1. *Weights of the criteria obtained from the first of the three DMs*

	Criteria	s_j	k_j	q_j	w_j
C_1	Requirement analysis		1	1	0.29
C_2	Status of the development team	0.85	1.15	0.87	0.25
C_3	User's participation	0.85	1.15	0.76	0.22
C_4	Project type and associated risk	1.1	0.9	0.84	0.24
				3.47	1.00

Source: Author's calculations

Table 2. *Weights of the criteria obtained from the second of the three DMs*

	Criteria	s_j	k_j	q_j	w_j
C_1	Requirement analysis		1	1	0.28
C_2	Status of the development team	0.89	1.11	0.90	0.26
C_3	User's participation	0.9	1.1	0.82	0.23
C_4	Project type and associated risk	0.98	1.02	0.80	0.23
				3.52	1.00

Source: Author's calculations

Table 3. *Weights of the criteria obtained from the third of the three DMs*

	Criteria	s_j	k_j	q_j	w_j
C_1	Requirement analysis		1	1	0.32
C_2	Status of the development team	0.7	1.3	0.77	0.24
C_3	User's participation	0.9	1.1	0.70	0.22
C_4	Project type and associated risk	1	1	0.70	0.22
				3.17	1.00

Source: Author's calculations

The group weights of the criteria based on the stances of the three DMs are shown in Table 4.

Table 4. *The weights of the criteria obtained from the three DMs*

	Criteria	w_j^1	w_j^2	w_j^3	w_j^*	w_j
C_1	Requirement analysis	0.289	0.284	0.316	0.296	0.296
C_2	Status of the development team	0.251	0.256	0.243	0.250	0.250
C_3	User's participation	0.218	0.232	0.221	0.224	0.224
C_4	Project type and associated risk	0.242	0.228	0.221	0.230	0.230
					0.999	1.000

Source: Author's calculations

Step 2. Ranking of alternatives

Based on the ratings obtained from the three *DMs*, group ratings are calculated as follows:

$$x_{ij} = \left(\prod_{k=1}^{3} x_{ij}^{k} \right)^{1/3}, \tag{7}$$

The group ratings of the four evaluated alternatives obtained from the three DMs are shown in Table 5.

Table 5. *The initial decision-making matrix*

Criteria Alternatives	C_1	C_2	C_3	C_4
w_j	0.296	0.250	0.224	0.230
A_1	3.33	3.00	2.67	2.33
A_2	2.67	2.67	2.67	3.33
A_3	3.67	3.67	3.00	4.00
A_4	5.00	4.67	4.67	4.67

Source: Author's calculations

By applying Eq. (2), a normalized decision matrix has been formed. The normalized decision matrix, as well as the weights of the criteria are shown in Table 6.

Table 6. *The normalized decision matrix and the weight of the criteria*

Criteria **Alternatives**	C_1	C_2	C_3	C_4
w_j	0.296	0.250	0.224	0.230
A_1	0.67	0.60	0.53	0.47
A_2	0.53	0.53	0.53	0.67
A_3	0.73	0.73	0.60	0.80
A_4	1.00	0.93	0.93	0.93

Source: Author's calculations

The relative importance of the evaluated alternatives, based on weighted sum (WS) method and weighted product (WP) are shown in Table 7.

Table 7. *The relative and total importance of the alternatives*

	$Q_i^{(1)}$	$Q_i^{(2)}$	Q_i	**Rank**
A_1	0.29	0.07	0.36	**3**
A_2	0.28	0.07	0.35	**4**
A_3	0.36	0.09	0.45	**2**
A_4	0.48	0.12	0.60	**1**

Source: Author's calculations

Data from the Table 7 show us that alternative designated as A_4 has the highest total importance in terms of evaluated criteria.

CONCLUSION

The pace of change in the software development industry is still high. People continue to push the limits of known techniques and practices to develop the most efficient and effective software. Software development lifecycle models and business decision models contribute to controlling product development in different ways.

A particular software development model can significantly affect various software product-related issues. If the model fails to fully meet the requirements, it will certainly affect the end product. Often a major reason for the failure of software development is the lack of good methodology or the implementation of inadequate. Also, a common barrier to successful software development is the misunderstanding and failure to meet user requirements. Continuous communication with the client is implied in agile methodologies, and such omissions are much harder to come by. Certainly, the ultimate goal, for both sides, is applicable software.

The proposed PIPRECIA-WASPAS approach has successfully responded to the requirements in terms of selection of the of software development methodology. The conducted case study has proved the applicability, ease of use and effectiveness of the proposed approach. Based on the conducted case study, alternative designated as A_4 has the highest total importance in terms of evaluated criteria. Therefore, SCRUM methodology is the most convenient by the stances of the DMs.

REFERENCES

Antović, I., Savić, D., Stanojević, V., Milić, M., & Vlajić, S. (2008). *Alati i metode softverskog inženjeringa po swebok projektu*. YU INFO - simpozijum 0 racunarskim naukama i informacionim tehnologijama. Kopaonik, 9-12.03.2008.

Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Managemen*t, 2(1), 26-30.

Baušys, R., & Juodagalvienė, B. (2017). Garage location selection for residential house by WASPAS-SVNS method. *Journal of Civil Engineering and Management*, 23(3), 421-429.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, (5), 61-72.

Buckley, F. J. (1987). An overview of the IEEE computer society standards process. *Computer Standards & Interfaces*, 6(2), 267-274.

Chakraborty, S., & Zavadskas, E. K. (2014). Applications of WASPAS method in manufacturing decision making. *Informatica*, 25(1), 1-20.

Cowley, B. (2014). *The QUARTIC process model for developing serious games:'green my place'case study*. In Digital Da Vinci (pp. 143-172). Springer, New York, NY.

Fliger, Š. L., Atli, D. M., Jovanović, S., Gonda, J., & Janković, R. (2006). *Softversko inženjerstvo: teorija i praksa*. Računarski fakultet.

Hernández, T. R., Kreye, M., & Eppinger, S. (2019). *Applicability of Agile and Scrum to Product-Service Systems*. In EurOMA Conference.

http://www.link-university.com/lekcija/Fazni-razvoj-i-spiralni-model/2620 (15.09.2019).

https://www.slideshare.net/Ehtesham17/waterfall-model-in-software-engineering (15.09.2019).

Jovanović, A. D., Jovanović, F. P., Miletić, L. Z., & Berić, I. M. (2016). Application of agile methodologies in software development. *Tehnika*, 71(6), 896-900.

Karabašević, D., & Maksimović, M. (2018). *The importance of the organizational learning's dimensions on the ability of innovation in the* *organization*. MEFkon 2018, Innovation as an initiator of the development –innovations – basis for development International Thematic Monograph – Thematic Proceedings, December 6th, Belgrade.

Karabašević, D., Stanujkić, D., Urošević, S., & Maksimović, M. (2016). An approach to personnel selection based on Swara and Waspas methods. *Bizinfo (Blace) Journal of Economics, Management and Informatics*, 7(1), 1-11.

Kilibarda, G., Šobajić, V., Berić, I., & Jovanović, P. (2016). Upravljanje softverskim projektima. *Tehnika*, 1, 145-153.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6), 47-56.

Mahapatra, H. B., & Goswami, B. (2015). *Selection of software development methodology (SDM): a comparative approach*. International Journal of Advanced Research in Computer Science and Software Engineering, 5(3), 58-61.

Manger, R. (2012). *Softversko inzenjerstvo*. Sveucilište u Zagrebu, PMF Matematicki odsjek, Zagreb.

Martin, R. C. (2002). *Agile software development: principles, patterns, and practices*. Prentice Hall.

Mead, N. R. (2009). *Software engineering education: How far we've come and how far we have to go*. Journal of Systems and Software, 82(4), 571-575.

Neill, C. J., & Laplante, P. A. (2003). *Requirements engineering: the state of the practice*. IEEE software, 20(6), 40-45.

Pfleeger, S. L., & Atlee, J. M. (1998). *Software engineering: theory and practice*. Pearson Education India.

Pichler, R. (2010). *Agile product management with scrum: Creating products that customers love*. Pearson Education India.

Royce, W. (1970, August). *The software lifecycle model (Waterfall Model)*. In Proc. WESTCON (Vol. 314).

Stanojević, V., Antović, I., Vlajić, S. (2006). *Kvalitet softvera po Swebok projektu*. Zbornik Radova, YU info.

Stanujkic, D., Karabasevic, D., Smarandache, F., Zavadskas, E. K., & Maksimovic, M. (2019). An Innovative Approach to Evaluation of the Quality of Websites in the Tourism Industry: a Novel MCDM Approach Based on Bipolar Neutrosophic Numbers and the Hamming Distance. *Transformations in Business and Economics*, 18(1), 149-162.

Stanujkic, D., Karabasevic, D., Zavadskas, E. K., Smarandache, F., & Brauers, W. K. (2019). A Bipolar Fuzzy Extension of the MULTIMOORA Method. *Informatica*, 30(1), 135-152.

Stanujkic, D., Zavadskas, E. K., Smarandache, F., Brauers, W. K., & Karabasevic, D. (2017b). A neutrosophic extension of the MULTIMOORA method. *Informatica*, 28(1), 181-192.

Stanujkic, D., Zavadskas, E. K., Karabasevic, D., Smarandache, F., & Turskis, Z. (2017a). The use of Pivot Pair-wise Relative Criteria Importance

Assessment method for determining weights of criteria. *Romanian Journal of Economic Forecasting*, 20(4), 116-133.

Stanujkić, D., & Karabašević, D. (2018). An extension of the WASPAS method for decision-making problems with intuitionistic fuzzy numbers: A case of website evaluation. *Operational Research in Engineering Sciences: Theory and Applications*, 1(1), 29-39.

Steward, D. V. (1987). *Software engineering with systems analysis and design*. PWS Publishing Co.

Stojić, G., Stević, Ž., Antuchevičienė, J., Pamučar, D., & Vasiljević, M. (2018). A novel rough WASPAS approach for supplier selection in a company manufacturing PVC carpentry products. *Information*, 9(5), 121.

Tomašević, V. (2012). *Razvoj aplikativnog softvera.* Univerzitet Singidunum, Beograd.

Turskis, Z., Zavadskas, E. K., Antucheviciene, J., & Kosareva, N. (2015). A hybrid model based on fuzzy AHP and fuzzy WASPAS for construction site selection. *International Journal of Computers Communications & Control*, 10(6), 113-128.

Urosevic, S., Karabasevic, D., Stanujkic, D., & Maksimovic, M. (2017). An approach to personnel selection in the tourism industry based on the SWARA and the WASPAS methods. *Economic Computation & Economic Cybernetics Studies & Research*, 51(1).

Zavadskas, E. K., Antucheviciene, J., Hajiagha, S. H. R., & Hashemi, S. S. (2014). Extension of weighted aggregated sum product assessment with interval-valued intuitionistic fuzzy numbers (WASPAS-IVIF). *Applied soft computing*, 24, 1013-1021.

Zavadskas, E. K., Turskis, Z., & Antucheviciene, J. (2015). Selecting a contractor by using a novel method for multiple attribute analysis: Weighted Aggregated Sum Product Assessment with grey values (WASPAS-G). *Studies in Informatics and Control*, 24(2), 141-150.

Zavadskas, E. K., Turskis, Z., Antucheviciene, J., & Zakarevicius, A. (2012). Optimization of weighted aggregated sum product assessment. *Elektronika ir elektrotechnika*, 122(6), 3-6.

View publication stats