IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 October 2024, accepted 14 November 2024, date of publication 18 November 2024,
date of current version 29 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3501672

RESEARCH ARTICLE

Using Axiomatic Design and Fuzzy Axiomatic Design for Risk Management in Software Development

TANER BEHÇET KEPKEP[1], ALPTEKİN DURMUŞOĞLU[2], AND TÜRKAY DERELİ[3]
[1]Engineering Faculty, Department of Industrial Engineering, Gaziantep University, 27310 Gaziantep, Türkiye
[2]Engineering Faculty, Department of Industrial Engineering, Samsun University, 55080 Samsun, Türkiye
[3]Office of President, Hasan Kalyoncu University, 27410 Gaziantep, Türkiye

Corresponding author: Taner Behçet Kepkep (tanerkepkep@gmail.com)

ABSTRACT The software development process is a type of structural approach, also called the software development life cycle, that includes many different steps including planning, requirements analysis, design, implementation, testing, deployment and maintenance, and guides the developer team from start to finish, covering the stages to meet the end user's needs and quality standards. The choice between agile and traditional software development life cycle (SDLC) methods significantly impacts the software development process, and developers must carefully consider which method to use to achieve a high-quality and sustainable end product. There are numerous SDLC models available and project managers and team members often select a model based on past experience rather than logical and rational decision-making process that can result in negative consequences, including software failures and budget overruns. To address these challenges, we chose to compare method selection between traditional and agile software development methodologies using Axiomatic Design (AD). AD provides a systematic and structured approach that takes into account the independence of functional requirements and allows for an explicit and mathematical evaluation of the properties of alternatives in decision problems. Our paper presents an objective and mathematical roadmap for selecting the appropriate SDLC model based on AD principles.

INDEX TERMS Axiomatic design, decision making, fuzziness, software development life cycle models.

I. INTRODUCTION

The software development process is not only a coding process but also an integrated production process with different phases. The process of software products being developed in a certain systematic stage is called the software development life cycle and includes planning, development, testing, deployment, and maintenance processes. There are lots of different types of SDLC methods that are convenient for different size and complexity software projects. SDLC helps project developers to design to improve the efficiency and effectiveness of the software development process. The most effective features of SDLC are reducing the risks, increasing efficiency, quality of production, and customer satisfaction. SDLC reduces the risks to be encountered at the beginning of the project with comprehensive planning, design, testing, and feedback. In terms of increasing efficiency, SDLC evaluates the software development phase in an organized and manageable way so that the project can be completed on time and within the budget. SDLC helps to increase the quality of the software developed with the test phase and feedback.

There are different SDLCs used during the development of software projects. The literature often categorizes SDLCs as either heavyweight or lightweight, or as traditional or agile methodologies. The choice of method depends on the size, scope, and capacity of the software project. Some of the most commonly used methods include Waterfall, Iterative, Spiral, Prototyping, Rapid Application Development (RAD), V-Model, Scrum, Extreme Programming (XP), Clean Room, Dynamic Systems Development, Rational Unified Process, Lean Software Development, Test-Driven Development, Feature-Driven Development, Model-Driven

The associate editor coordinating the review of this manuscript and approving it for publication was Wojciech Sałabun.

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

Engineering, Crystal Methods, Joint Application Development, Adaptive Software Development, and Open Source Software Development. Each of these methods possesses distinct characteristics, strengths, and weaknesses. Our research aims to identify the most suitable SDLC method for developing a software project. The literature includes studies that highlight the significance of SDLC and the appropriateness of various methods for specific software projects [5], [17], [20].

In this paper, we consider the determination of the appropriate SDLC for a software project to be developed as a decision problem. The reason for this is the factors that will be effective while developing the project; these can be listed as the technical competence of the team that will develop the software, the size of the project, and the impact of the software requestor on the project. In the literature, multi-criteria decision-making methods (MCDM) have been used in the solution of decision problems, and we consider our research as a decision problem.

Among the most commonly used decision-making methods are Analytic Hierarchy Process (AHP), Analytic Network Process (ANP), Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE), ÉLimination Et ChoixTraduisant la REalité (ELECTRE), and Fuzzy decision-making methods. Apart from these frequently used methods, axiomatic design is also used for decision-making methods. In this paper, unlike other studies, we prefer to use the axiomatic design method in the decision problem. The main reason for using AD is that it offers a structured approach to design problems. Its feature is to help identify the functional requirements of a design and ensure that they are independent of each other. Axiomatic design leads to a more robust and efficient result that is less likely to fail or require costly changes. AD helps decision-makers reduce the complexity of a design problem by breaking it down into smaller, more manageable segments. Overall, axiomatic design is a powerful tool for improving the quality and efficiency of design processes in a variety of areas.

II. LITERATURE REVIEW

The usage of appropriate software development life cycle models is essential for both software planning and development. There are numerous academic studies and approaches in the literature that employ MCDM models to demonstrate the relevance of selecting convenient SDLC models. The goal of each research is to select the appropriate SDLC models using the MCDM approach, a crucial step for both the software development team and the final products produced. Numerous studies indicate that the development process is influenced by the structure of the development team, the type of project, and the organizational structure. Each of these structures directly affects the process of ensuring the robustness and quality of the software products. In their research, Vijayasarathy and Butler (2016) mention that project type, size, team structure, and team abilities affect the software products. The authors mention that SDLC method selection should be determined as a complement and not as a competition because the criteria to be compared are considered compatibility and not comparison and they claim that the structure and capability of each SDLC method should be analyzed to assess its suitability for the software project to be developed [30]. Coleman and O'Connor seek an answer to the question in their research, ''Which SDLC will be suitable for the project?'' asked by the software developer [8]. The authors took into account the characteristics and capacities of SDLC in their research. In the research, authors evaluate the situational factors that affect the software development process and establish a structural framework that aims to provide guidance to software developers [8]. Sheffield and Lemétayer's empirical research article on successful projects aims to answer the question: Which factors in the project and its environment are indicators of software development agility in successful projects? [26] In order to determine which SDLC would be best for developing software with a variety of features, sizes, and strategies, Soobia et al. analyzed the objectives, capacities, and characteristics of each SDLC [27]. Vijayasarathy and Butler address in their study whether the organization, the structure of the project, and the work team have an impact on the project and the software development process [30]. Many studies have explored not only the significance of Software Development Life Cycles (SDLCs) for software projects but also the selection of the most appropriate SDLC method. The selection of a suitable SDLC has been considered a decision problem in many studies. As is well known, MCDM methods are one strategy to use for decision problems where the criteria overlap and there are multiple alternatives. MCDMs, in addition to software methodology selection, have found application in various computer sciences areas to resolve decision problems. Another study employs MCDM methods to select ERP alternatives [9]. In another study, Riberio et al. aim to evaluate the suitability of MCDM methods to support software engineers' decisions during software development [25]. Different fields of computer science have used MCDMs for decision-making. Additionally, we consider the selection of the software development life cycle method as a decision problem and strive to find the appropriate method for the project. In addition to MCDM methods, different approaches have been used to select the appropriate SDLC for a software project. Ahmar uses criteria such as project duration, clarity of user requirements, and familiarity with the technology, system complexity, system reliability, and schedule visibility to determine the appropriate SDLC method with the help of an expert system [1]. In the literature, Gaur and Aggarwal mention the use of the wrong SDLC as one of the reasons for failed software, and they use the TOPSIS method to select the appropriate SDLC. Decision-making methods have been developed for different types of decision problems. The structure of these decision problems may be different from each other and may not be suitable for every decision problem. Among the most widely used methods in

the literature are those such as AHP, ANP, Topsis, Electre, Promethee, and fuzzy approaches. Fuzzy approaches aim to make more accurate decisions by using fuzzy numbers. The software industry is one area where MCDM methods can find application. Many areas in the software industry use MCDM methods, and the process of creating useful, long-lasting, and improvable software defines the use of MCDM as an important decision problem. In addition to good coding, a software development process requires good planning and evaluation of external factors, and this is possible by selecting the appropriate SDLC for the project being developed. In this paper, we aim to identify the most efficient SDLC method for the development duration, utilizing both axiomatic and fuzzy axiomatic designs.

III. METHODOLOGY

Axiomatic Design (AD), developed by Suh (1998), is a design method that aims to answer the questions ''what do we want to achieve and how can we achieve it'' and is intended to bring more scientific outcomes to production and design processes. In addition to being a design method, AD is also utilized in literature for decision problems. The main goal of AD is to establish a solid foundation for design and achieve results by logically supporting processes and tools.

AD can be defined as a logical and methodical approach to designing a product or system. The process, which is like solving a mathematical problem, involves certain rules and axioms. This makes the design process more organized and understandable. Two fundamental principles, independence and information axioms, form the foundation of AD design. The independence axiom states that design components are independent. Any change to one component does not affect the others, making the design more adaptable and versatile. For instance, we can define the battery life of smart phones and screen size as independent axioms. To clarify, neither the screen size nor the battery life has any impact on each other. Such design axioms are independence axioms because they do not affect each other. We use the information axioms when the information content is minimal. Too much information increases the complexity and makes the design difficult to understand. Therefore, the design includes only the necessary features. As an illustration of the information axiom, choosing a design for smart phones that utilizes fewer components or simpler materials can streamline the production process and minimize the likelihood of failures. Among the advantages of AD are its step-by-step and logical design advancement, and its flexibility makes it easier to make modifications, thereby improving design quality and speeding up the design process.

With these characteristics, AD is a powerful method used in system and product design, allowing for the development of designs in a logical and methodical manner while focusing on the needs. An AD flowchart can be defined as follows: The first step is identification of needs, which means determining by whom and for what purpose the product will be used. The second step is the determination of functional requirements; what the product should do, i.e., which functions it will fulfil, is defined in detail. The following step is the determination of design parameters, which determine which features the product must have in order to fulfil its functions. The fourth step involves the determination of process variables, and it is planned how the product will be produced according to the determined features. The last step is evaluation of the design; the conformity of the design to the initial axioms and the specified criteria is checked. Figure 1 shows the flowchart of AD.

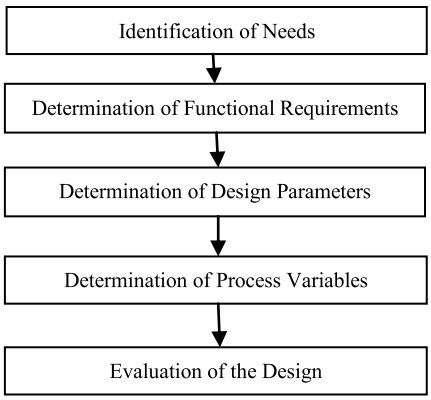

FIGURE 1. Flowchart of axiomatic design.

AD design addresses two different axioms, the first of which is the independence axiom, which argues that the independence of functional requirements (FR), defined as the minimum number of independent functional needs that characterize the design objectives, should be continuously maintained. When more than one FR arises, the design solution should fulfill each functional need without affecting the other functional need. This means choosing the right set of design parameters that fulfill the functional needs and maintain their independence. Another axiom is the information axiom, which argues that among the designs that satisfy the independence axiom, the design with the minimum information content is the best design. This is because the information content is defined in probability terms, and at the same time, according to the second axiom, the design with the highest probability of realization is the best design.

The selection of SDLC methods with AD, which is the main objective of our research, is illustrated by the flowchart in Figure 2. The independence and information axioms, which form the basis of AD, show the process of SDLC selection step by step and provide a guideline, thus facilitating the decision-making process. The flowchart provides a visual representation of the principal decision points and processes involved in applying AD to the selection of an SDLC model.

The information content I is defined by the probability of achieving a given FR. If the probability of success of the probability of achieving a given FR is p, the information content I related to the probability is denoted by Equation 1.

$$I_i = log_2 \frac{1}{p_i} \tag{1}$$

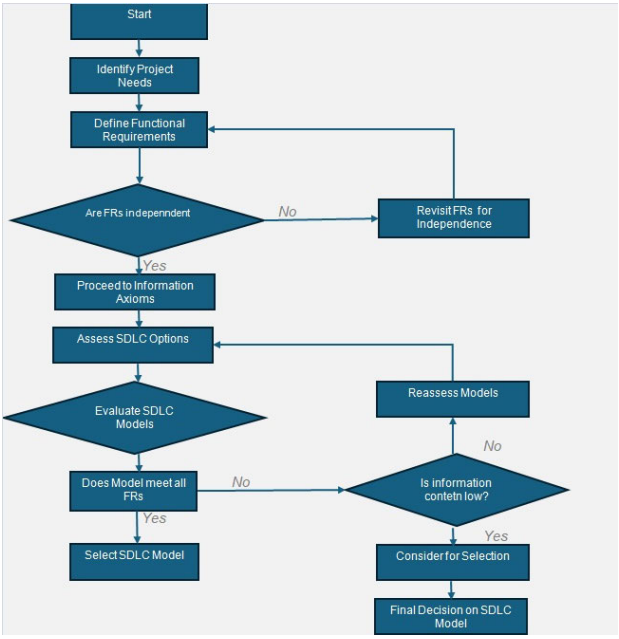

FIGURE 2. Flowchart of Illustrating the application of AD principles in SDLC selection.

Information is given in small units. Since there are n FR s, the total information content is the sum of all these possibilities. When the sum of all probabilities is equal to 1, the information content is zero, and conversely, when one or more probabilities are equal to zero, the required information is infinite. If this probability is low, it means that more information is needed to meet the functional needs [27].The probability of realization can be calculated by specifying the Design Range (Dr) for the FR and the System Range (SR) for the design that will achieve the FR. Figure 3 shows that when the system probability distribution function of an FR is uniform, the intersection of the 'design range' specified by the designer and the 'system range' realized by the system is the area where an acceptable solution is found. When the system probability distribution function is uniform, the probability of FR is calculated by Equation 2:

$$p_i = \frac{CommonRange}{SystemRange} \tag{2}$$

Based on Equation 2, the information content is calculated as Equation 3:

$$I_i = \log_2 \left(\frac{SystemRange}{CommonRange}\right) \tag{3}$$

If FR_i is a continuous random variable, the probability of realizing FR_i in the design interval is calculated by Equation 4, where $P_s\ (FR_i)$ is the system probability density function for each FR_i; Equation 4 gives the probability of realizing the interval of the whole system by taking the integral of the probability density function of the system.

$$Pi = \int_{dr^l}^{dr^u} P_s\ (FR_i)\, dFr_i \tag{4}$$

Figure 4 shows a probability density function against the FI for which the system range is defined. The intersection region between the design range and the system range is denoted as the common range (Cr), which is the region where only functional requirements are fulfilled. Consequently, the area under the system range divided by the area under the common range is equal to the probability of the degree of realization of the specified objective of the design.

$$I = \log_2 \left(A_{sr} / A_{cr}\right) \tag{5}$$

where A_{sr} is the area under the system range, A_{cr} is the common range refers to the hatched area below the range. Usually A_{sr} Since there are n FRs to be provided since $=1.0$ The information content is expressed by Equation 6:

$$I = \log_2 \left(1 / A_{cr}\right) \tag{6}$$

In literature, most of the MCDM methods use certain data to get a result. Fuzzy Axiomatic Design is a method that can be used when the data are not certain. When the data are certain, numerical values can be used to represent the values. However, data are not represented by numerical values; linguistic variables can be used to represent values. Fuzzy set theory is an important tool in this stage. Figure 5 shows the membership function of nonnumeric factors converted to numerical values. In axiomatic design, the system and design ranges of functional requirements are not always expressed by a specific range. They can be expressed above a certain value or approximated to a value, and these values can be represented by triangular or trapezoidal fuzzy numbers. In Fuzzy Axiomatic Design, triangular or trapezoidal fuzzy membership functions are used when interval values are given linguistically and the probability density function is certain. This allows the common area to be represented by triangular or trapezoidal fuzzy numbers in the intersection region. As shown in Figure 6, the common area is the intersection between the fuzzy triangular field of the system range and the fuzzy triangular area of the design interval.

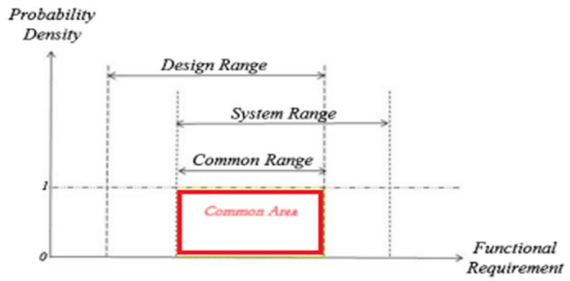

FIGURE 3. Design range, system range, common range and probability density function of a functional requirement.

In this way, the information content is calculated as in Equation 7.

$$I = \log_2 \left(\frac{TFNofSystemRange}{CommonArea}\right) + \log_2 \left(\frac{TFNofDesignRange}{Common}\right) \tag{7}$$

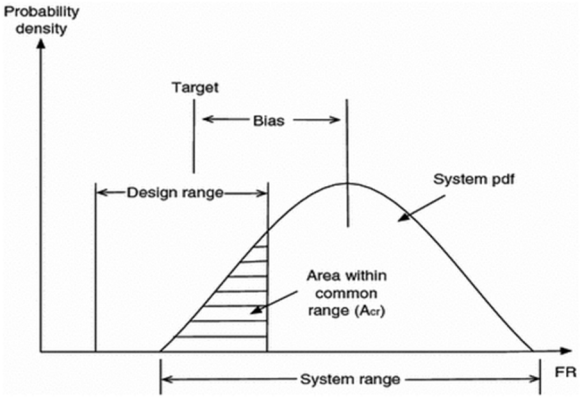

FIGURE 4. Design range, system range, common range and probability density function of a functional requirement (FR).

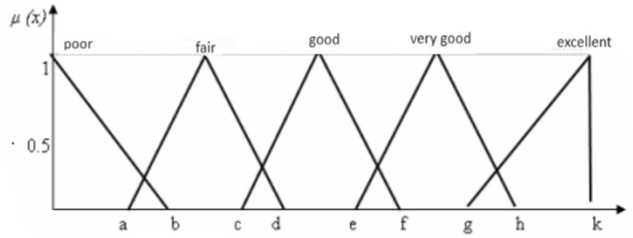

FIGURE 5. Numerical representations for intangible factors.

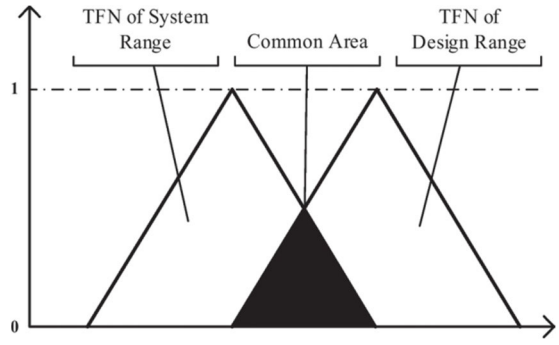

FIGURE 6. Common Range of system and design range.

The weighted fuzzy axiomatic design is evaluated differently from the previous calculations. In the weighted fuzzy axiomatic design, the weights of the criteria are not considered equal and the calculation shown in Equation 8 is used in addition to the current calculation.

$$\begin{cases} \left[\log_2\left(\frac{1}{P_{ij}}\right)\right]^{1/w_j}, & 0 \le I_{ij} \le 1 \\ \left[\log_2\left(\frac{1}{P_{ij}}\right)\right]^{w_j}, & I_{ij} \ge 1 \\ w_j, & I_{ij} = 1 \end{cases} \tag{8}$$

IV. DETERMINING THE BEST FIT SDLC METHODOLOGY FOR A SOFTWARE PROJECT

Software development is the set of systematic steps followed to transform an idea into software, which is a complex process in which many different phases take place and includes many important and effective steps. Software development comprises several interdependent parts each with its own set of challenges and objectives. The phases of the software development process are depicted in Figure 7.

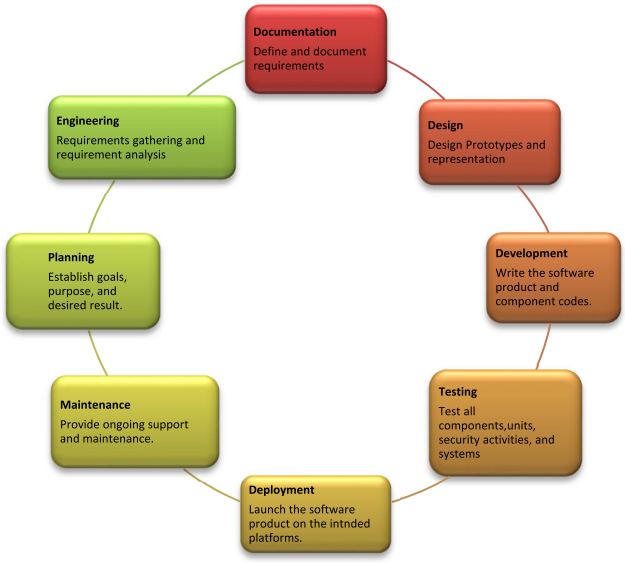

FIGURE 7. Flowchart of software development life cycle.

Many studies in the literature describe software methodologies and their characteristics. In these researches, both traditional and agile software methodologies are mentioned. In addition to the technical features of the methodologies, managerial and customer orientated features were also mentioned in the researches. Due to these different characteristics, determining which methodology is appropriate to use during a project to be developed is considered a decision problem in our study. The research highlights the unique characteristics of each methodology. Many studies on software development models have mentioned the strengths and weaknesses, as well as the advantages and characteristics of the methods. In addition to the importance of software development methodologies, various factors affecting the software development process have also been the subject of research. Researchers have emphasized that the working environment of software developers, emotional factors, and the guidance of project managers responsible for controlling the software process are among these factors that influence the process [7], [10], [31]. Islam and Ferworn discussed the general features of SDLC methods, dividing them into two main categories: traditional and agile methodologies [17]. These methods consider features such as the type of approach, project size, management style, documentation, emphasis, team size, cost, and so on [17]. In another article, researchers use the same features to clarify and define SDLC methods [20]. Misra et al., in their research, pointed to more personal and technical factors when choosing agile software development life cycle models. The authors mentioned more personal features

like customer satisfaction, customer collaboration, customer commitment, communication and negotiation, social culture, personal characteristics, and more [21]. Hicdurmaz categorized the software development process into three main parts. Hicdurmaz identified three main components of the software development process: the technical features, which include requirements, testing, design, and architecture; the process part, which involves complexity, criticality, flexibility, reusability, and quality; and the personal features, which involve ease of management, user involvement and feedback, and cost [16]. Numerous studies in the literature mention cost, complexity, resource requirements, customer involvement, documentation, testing, risk management, flexibility, and quality as characteristics of the methods [4], [10], [12], [19], [23], [24], [31]. Academic studies in the literature mention not only the technical characteristics of software development methodologies, but also personal and managerial aspects. Therefore, we will discuss both traditional and agile methods in our study, taking into account the needs of developers and managers. The characteristics of the methodologies will determine the necessary criteria for stakeholders. The stakeholders will select the characteristics of the methods, taking into account the capacities of the software development life cycle models. Note the characteristics of traditional and agile methods and how they are defined in the literature at this stage. Firstly, if we examine the waterfall model, which is the first of the traditional models, it is a sequential model that has an inflexible and rigid structure and requires each stage to be completed before moving to the next stage. The steps of the waterfall model include requirements gathering, design, implementation, testing, deployment, and maintenance, and it is difficult to incorporate changes after these phases are completed. The V Model is a different structure of the waterfall model and has a testing process corresponding to each stage. The importance of test steps is emphasized during the development process. Each cycle in the process represents a stage of the software development process. Another traditional model is the spiral model, which combines elements of the iterative development waterfall model and is a risk-oriented model. Each cycle in the process represents a stage of the software development process. The incremental model is the most flexible model among the other traditional models, and the process splits into smaller modules. Each of these modules developed during the process, and these help to allow incremental improvement and feedback. Because of the flexibility of the model, changes can be incorporated at each increment. One of another traditional model is the prototype model, which helps stakeholders to model is working and it helps to create a working model. The prototype model is usually used for gathering feedback and refining requirements before developing the final product. Prototyping helps us understand user needs and expectations early in the development process. Rapid Application Development (RAD) is a traditional method that emphasizes iteration and rapid development. It follows a sequential development process. The RAD methodology focuses a strong emphasis on producing prototypes rapidly, depending on user feedback, and with little before planning. To guarantee that the finished product fulfills user expectations, the RAD approach places a strong emphasis on user involvement throughout the development process. Agile software development life cycle models are iterative and incremental approaches to software development that emphasize flexibility, cooperation, and adaptability to change. Scrum is a prominent agile paradigm that focuses on cooperation, accountability, and iterative development. It entails breaking down the project into short, manageable sections called sprints, which typically run for 2–4 weeks. Scrum teams hold daily short meetings, sprint planning, sprint review, and sprint retrospective sessions to ensure continuous progress. Extreme Programming (XP) is a software development methodology that emphasizes providing high-quality software quickly and consistently. This strategy focuses on practices like test-driven development, pair programming, continuous integration, and frequent releases. XP encourages client engagement, simplicity, and adaptability to changing requirements. Kanban is a visual agile methodology that enables teams to manage their work by visualizing the process on a Kanban board. Work items are shown as cards that progress through the workflow, from ''To Do'' to ''Done.'' Kanban supports a pull-based system, in which work is moved to the next step only when capacity allows, resulting in smoother flow and reducing bottlenecks. Lean software development is based on the concepts of lean manufacturing and aims to provide value to customers with little waste. This strategy focuses on continual improvement, removing non-value-adding processes, and empowering teams to make decisions. Lean concepts include optimizing the entire process, establishing integrity, and reinforcing learning. The Crystal Model is a set of agile approaches that differ depending on project size, criticality, and team composition. Crystal methods promote communication, simplicity, and the frequent release of working software. Crystal methodologies are designed to meet the unique needs and peculiarities of each project.

These agile software development lifecycle methods, which embrace change and produce usable software in brief iterations, encourage cooperation, adaptation, and customer happiness. To produce software development outcomes that are successful, teams can select the agile model that best fits their organizational culture and project requirements. In the article, we discuss which software development methodology should be chosen for developing the software. The evaluations and classification were handled in this direction, and a choice was made between methodologies based on the opinion of experts. The verbal expressions of the expert opinions were translated into numerical expressions as shown in Figure 3. In our research, we will mention both traditional and Agile SDLC models as alternatives that are convenient for wished-developed software projects. The methodologies considered in the article are shown in Table 1. Software

Development Life Cycle methodology features will be used as criteria and these criteria are shown in Table 2. Software engineers and project managers will give their opinion about the methodologies features.

TABLE 1. Considered software development life cycle models.

Traditional Methodologies	Agile Methodologies
Waterfall	Scrum
V-Model	Extreme Programming (XP)
Spiral	Lean Software Development
Incremental	Kanban
Prototype	Crystal
Rapid Application Development	

In order to determine a framework or roadmap, criteria must be determined from the characteristics of the software development processes. These criteria will directly affect the success and functioning of the software to be developed. In the studies in the literature, researchers have addressed the characteristics that are especially effective in project development time. In their study, Chandi et al. made a selection among agile methods and in the research, delivery time, budget cost, time boxing, and response to change, documentation capacity, and project capacity were included as method features [6]. In another research article about deciding which agile methodology is the best for evaluating projects, additional considerations are made about development style, team size, team distribution, customer involvement, and iteration time period (Harb et al.). When the studies in the literature are examined in general, the criteria used in the determination of SDLCs are approximately the same [2], [3], [6], [13], [14], [15], [18], [19], [22], [23], [24], [28]. In the research, either traditional models or only agile models were evaluated. In our research, we address the characteristics of both agile and traditional methods.

V. APPLICATION OF FUZZY AXIOMATIC DESIGN FOR SDLC SELECTION

The research topic of our article will be the determination of the software development life cycle to be used in a software project developed for a corporate company. At this stage, the choice between the classical and agile software development life cycles will be determined by fuzzy axiomatic design and weighted fuzzy axiomatic design.The characteristics used as criteria during the determination of software development methods are shown in Table 4. With these criteria and the opinions of the software development team, SDLCs will be selected. In order to apply the axiom of information in axiomatic design, the independence axiom of axiomatic design must first be satisfied. In our study, the criteria, which are the functional requirements for alternatives, are independent of each other. In order to calculate the information content of each alternative, the design ranges of the functional requirements (FR) need to be determined. The design ranges of the criteria to be used during the study are shown in Table 2. The criteria we consider for method selection in the paper are flexibility, cost, duration, customer interaction, complexity, documentation, requirement, and project size, which are essential criteria for project development in our research. According to the project manager, software developers, and stakeholders, they have a common idea on the decision problem: what the criteria are and what the linguistic values are, and so the criteria and design range are shown in Table 2.

TABLE 2. Criteria and design range.

Criteria	Design Range
Flexibility	(2,8,14)
Cost	(5,14,14)
Duration	(5,14,14)
Requirement	(5,10,14)
Complexity	(3,7,14)
Project Size	(4,10,14)
Documentation	(3,10,14)
Customer Interaction	(3,14,14)

In Table 3 represent alternatives of SDLC models and Table 4 represent the criteria which will use in decision problem. Table 5 shows SDLC's to be considered in the project development and the evaluation of the necessary criteria for the project by the developers, project manager and stakeholder involved in the project.

TABLE 3. Abbreviations of SDLC model representations.

SDLC Models	Abbreviations
Waterfall	A1
V-Shape Model	A2
Spiral	A3
Incremental	A4
Prototype	A5
RAD	A6
Scrum	A7
XP	A8
Crystal	A9

TABLE 4. Abbreviation of features of SDLC models.

Features	Abbreviations
Flexibility	C1
Cost	C2
Duration	C3
Requirement	C4
Complexity	C5
Project Size	C6
Documentation	C7
Customer Interaction	C8

The capacities of the features of SDLCs mentioned in the literature research and the design range values with expert opinions are shown in Table 5. Using these design range values and fuzzy system domains, the appropriate software development method will be determined.

Using Equation 3 and Equation 5, the information content of each SDLC method can be calculated. We will illustrate the

TABLE 5. Linguistic evaluation of experts with respect to design range.

	C1	C2	C3	C4	C5	C6	C7	C8
A1	B	G	B	B	G	B	VG	B
A2	B	B	B	F	B	F	G	B
A3	G	B	B	F	B	G	VB	G
A4	F	F	F	B	B	G	F	F
A5	G	B	F	B	F	G	F	G
A6	B	B	G	F	B	G	F	F
A7	F	F	G	B	F	F	B	F
A8	VG	F	VG	B	G	F	VB	G
A9	G	F	G	F	F	F	F	F
A10	G	F	G	B	B	G	F	F

Abbreviation of Linguistic VB: Very Bad, B: Bad, F: Fair, G: Good, VG: Very Good

solution path with the V-Model, one of the traditional models, and Extreme Programming, one of the agile models. In our paper, we used the Geogebra to achieve certain mathematical results and to make the results more understandable and clear. The project cost design values for the V-Model are (5, 14, 14), while the project cost system value is evaluated as Bad.

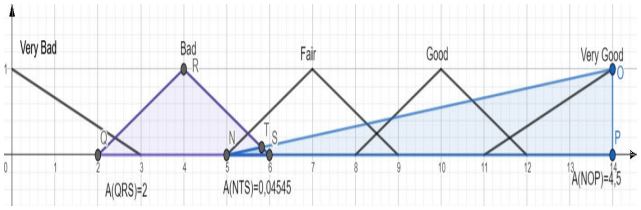

FIGURE 8. The common area of fuzzy system and design ranges of V-Model for project cost.

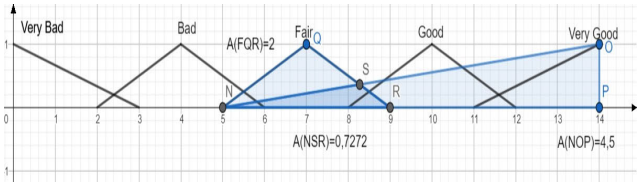

FIGURE 9. The Common area of fuzzy system and design ranges of extreme programming for project cost.

In Figure 8, the QRS triangle shows the system range of project cost, the NOP triangle the design range, and the NTS triangle the common area. When we include the area of the NTS triangle and the QRS triangle, which represent the common area, in the calculation as in equation 3, we obtain the value 5.4595.

$$I_i = \log_2\left(\frac{2}{0,\,04545}\right) = 5,4595$$

In Figure 9, the NQR triangle shows the system range of project cost of Extreme Programming, the NOP triangle the design range, and the NSR triangle the common area. When we include the area of the NQR triangle and the NSR triangle, which represent the common area of project cost, in the calculation as in equation 3, we obtain the value 5.4595.

$$I_i = \log_2\left(\frac{2}{0,\,7272}\right) = 1,4595$$

Similar calculations were made for other SDLCs used in our study, and the information contents determined for functional needs are given in Table 6. When the results shown in Table 6 are analyzed, the alternative with the lowest value in the last column showing the sum of the information content should be selected. Since design ranges are used in AD, an alternative must be within the design range. The most important feature of AD that distinguishes it from MCDM methods is that it does not accept an alternative if the design interval is not met for any criterion.

TABLE 6. The results of information contents for SDLC methods.

	I_{C1}	I_{C2}	I_{C3}	I_{C4}	I_{C5}	I_{C6}	I_{C7}	I_{C8}	$\sum I$
A1	1	0,3360	5,4595	4,8073	0,3219	3	1,6375	2,5303	19,0925
A2	1	5,4595	5,4595	0,8074	1,4146	0,4764	1	2,5303	18,1477
A3	0,1926	5,4595	5,4595	0,8074	1,4146	0	Infinity	0,2125	Infinity
A4	0,0458	1,4595	1,4595	4,8073	1,4146	0	0,3219	0,7828	10,2914
A5	0,1926	5,4595	1,4595	4,8073	0	0	0,3219	0,2125	12,4533
A6	1	5,4595	0,3360	0,8074	1,4146	0	0,3219	0,7828	10,1222
A7	0,0458	1,4595	0,3360	4,8073	0	0,4764	1,9995	0,7828	9,9073
A8	1,5849	1,4595	0	4,8073	0,3219	0,4764	Infinity	0,2155	Infinity
A9	0,1926	1,4595	0,3360	4,8073	1,4146	0	0,3219	0,7828	9,3147

In Table 6, the Crystal method with a total information value of 9.3147, which has the smallest value, is the most appropriate alternative among the alternatives.

VI. THE APPLICATION OF WEIGHTED FUZZY AD APPROACH OF SDLC SELECTION

In the decision-making process, if the criteria are not of equal importance and their degrees of importance are different from each other, the weighted fuzzy AD approach can be used. At this stage, the first thing to be done is to determine the weight values of the criteria. In our article, the 1-9 importance scale developed by Saaty is used, and the weights of the criteria are calculated by analyzing the pair wise comparison matrix with the eigenvector method proposed by Saaty. The weight values calculated with the pair wise comparison matrix are shown in Table 7. Using Equation 8, the information content for the Waterfall method, which is one of the alternative methods, is calculated as shown in Table 8.

TABLE 7. Pair wise comparison matrix for the criteria and their weights.

	C1	C2	C3	C4	C5	C6	C7	C8	Weight(w)
C1	1	1\5	5	4	1	1\6	5	3	0,227
C2	5	1	1	3	3	3	5	3	0,287
C3	1\5	1	1	3	3	3	5	3	0,230
C4	1\4	1\3	1\3	1	1\5	5	2	3	0,119
C5	1	1\3	1\3	5	1	5	6	3	0,202
C6	6	1\3	1\3	1\5	1\5	1	5	5	0,173
C7	1\5	1\5	1\5	1\2	1\6	1\5	1	1\3	0,032
C8	1\3	1\3	1\3	1\3	1\5	1\5	3	1	0,059

TABLE 8. Calculation of information contents for the waterfall model using axiomatic design equations.

A1	I_{C1}	1	$w_j, I_{ij} = 1$	$\rightarrow 0{,}227$
	I_{C2}	0,3360	$\left[\log_2\left(\frac{1}{P_{ij}}\right)\right]^{1/w_j} 0 \leq I_{ij} \leq 1$	$\rightarrow 0{,}02236$
	I_{C3}	5,4595	$\left[\log_2\left(\frac{1}{P_{ij}}\right)\right]^{w_j} I_{ij} \geq 1$	$\rightarrow 1{,}4775$
	I_{C4}	4,8073		$\rightarrow 1{,}2054$
	I_{C5}	0,3219		$\rightarrow 0{,}0036$
	I_{C6}	3		$\rightarrow 1{,}2093$
	I_{C7}	1,6375		$\rightarrow 1{,}0159$
	I_{C8}	2,5303		$\rightarrow 1{,}0562$
	$\sum I$			6,21726

TABLE 9. The result obtained with weighted fuzzy AD.

w	0,227	0,287	0,230	0,119	0,202	0,173	0,032	0,059	
	I_{C1}	I_{C2}	I_{C3}	I_{C4}	I_{C5}	I_{C6}	I_{C7}	I_{C8}	$\sum I$
A1	0,227	0,022 36	1,47 75	1,20 54	0,00 36	1,20 93	1,01 59	1,0562	6,217 26
A2	0,227	1,627 6	1,47 75	0,16 56	1,07 25	0,01 37	0,03 2	1,0562	5,672 1
A3	0,0007	1,627 6	1,47 75	0,16 56	1,07 25	0	infinity	0,000000 04	Infinity
A4	0,0000 01	1,114 6	1,09 08	1,20 54	1,07 25	0	0	0,0157	4,499 0
A5	0,0007	1,627 6	1,09 08	1,20 54	0	0	0	0,000000 04	3,924 5
A6	0,227	1,627 6	0,00 87	0,16 56	1,07 25	0	0	0,0157	3,117 1
A7	0,0000 01	1,114 6	0,00 87	1,20 54	0	0,01 37	1,02 24	0,0157	3,380 5
A8	1,110	1,114 6	0	1,20 54	0,00 36	0,01 37	Infinity	0,000000 04	infinity
A9	0,0007	1,114 6	0,00 87	1,20 54	1,07 25	0	0	0,0157	3,417 6

When the results obtained in Table 9 are analyzed, it is seen that the alternative with the least information content is the RAD model indicated with A6. The results obtained with fuzzy AD and weighted fuzzy AD methods are shown in Table 10. When Table 10 is analyzed, it is seen that the most appropriate alternative SDLC model ranking that can be used during project management has changed. It is clear from this result that taking into account the criteria weights has a significant effect on the result. The Spiral model and XP model, which were among the alternatives, were eliminated from the alternatives because they did not fully meet the documentation requirement.

TABLE 10. he result obtained with fuzzy AD and weighted fuzzy AD approaches.

SDLC Methods	Fuzzy Axiomatic Design		Weighted Fuzzy Axiomatic Design	
	$\sum I$	Priority	$\sum I$	Priority
A1	19,0925	7	6,21726	7
A2	18,1477	6	5,6721	6
A3	Infinity	Eliminated	Infinity	Eliminated
A4	10,2914	4	4,4990	5
A5	12,4533	5	3,9245	4
A6	10,1222	3	3,1171	1
A7	9,9073	2	3,3805	3
A8	Infinity	Eliminated	infinity	Eliminated
A9	9,3147	1	3,4176	2

VII. RESULT AND CONCLUSION

For a quality and successful software project, selecting the appropriate SDLC model is at the beginning of the project's development process. In general, due to the fact that the requirements, resource management, and other important criteria are not taken into consideration carefully during software development, strong, high-quality, and useful software cannot be obtained, causing loss of time, labor, and money. Generally, at the beginning of the software development process, the requirements are based on imprecise, uncertain data, so it is more appropriate to express alternatives with linguistic variables rather than precise numerical values. AD is important because it provides a systematic methodology for designing products, processes, projects, and systems by translating needs into functional requirements, design parameters, and process variables. In this study, AD technique is used, and different criteria are evaluated together. Accordingly, software project managers and software developers determined the design ranges for the functional requirements that the alternatives should fulfill. All criteria are expressed in linguistic variables, and triangular fuzzy numbers are used to express the system and design ranges numerically. In order to determine the SDLC model with the least information content, the system ranges realized by the alternatives for each criterion (Functional Requirement-FR) were determined, and the information content was calculated for each alternative. Then, with the same system and design intervals, the information contents were recalculated by considering the situation in which the criteria were weighted. It was observed that the results obtained were different in cases where the weights of the criteria were taken equal.

In addition to the theoretical advantages, an evaluation of the practical consequences of the research findings reveals that AD and fuzzy AD methods provide a structured decision-making framework. The framework enables project managers and developers to make more informed choices and ensures that the selected SDLC is compatible with the project size, complexity, and resource availability. The application of AD

and fuzzy AD, in the selection of an SDLC can be considered a practical decision-making tool for project managers and software developers. By adopting this approach, stakeholders are able to evaluate SDLC models in a systematic manner according to a range of criteria, including flexibility, cost, complexity, and customer interaction. This structured approach serves to minimize the likelihood of discordance between the project requirements and the selected SDLC, thereby facilitating the successful delivery of the project. During the software development process, there is a risk of utilizing an inappropriate SDLC, which could result in the production of a suboptimal product, accompanied by budgetary constraints. The selection of an unsuitable SDLC can lead to budgetary overspends or the delivery of low-quality software. By using fuzzy AD and weighted fuzzy AD frameworks, such negativities can be prevented, and a project that meets team expertise, project scope, and customer expectations can be provided with the appropriate SDLC selected. As a result of the research, project managers and software developers can evaluate the process more clearly. Project managers and software developers obtain a guide within the findings they obtain. Thanks to this guide, project requirements are defined, the most appropriate SDLC model is safely selected by applying the independent and information axioms, the risks to be encountered are reduced, and the project result is improved. The proposed approach enables more sustainable project management practices over time. With the proposed approach, organizations have a more adaptable and durable software development process. The proposed method brings long-term positive results such as project efficiency, customer satisfaction, and product quality, as well as the ability to make more rational and result-oriented decisions about SDLC selection.

REFERENCES

[1] M. A. A. Ahmar, "Rule based expert system for selecting software development methodology," Tech. Rep., 2005.

[2] J. E. T. Akinsola, A. S. Ogunbanwo, O. J. Okesola, I. J. Odun-Ayo, F. D. Ayegbusi, and A. A. Adebiyi, "Comparative analysis of software development life cycle models (SDLC)," in *Intelligent Algorithms in Software Engineering*, vol. 1224, R. Silhavy, Ed., Berlin, Germany: Springer, 2020, pp. 310–322, doi: 10.1007/978-3-030-51965-0_27.

[3] S. Alshehri, "Multicriteria decision making (MCDM) methods for ranking estimation techniques in extreme programming," *Eng., Technol. Appl. Sci. Res.*, vol. 8, no. 3, pp. 3073–3078, Jun. 2018, doi: 10.48084/etasr.2104.

[4] M. A. Ben-Zahia and I. Jaluta, "Criteria for selecting software development models," in *Proc. Global Summit Comput. Inf. Technol. (GSCIT)*, Jun. 2014, pp. 1–6, doi: 10.1109/GSCIT.2014.6970099.

[5] G. Büyüközkan, C. Kahraman, and D. Ruan, "A fuzzy multi-criteria decision approach for software development strategy selection," *Int. J. Gen. Syst.*, vol. 33, nos. 2–3, pp. 259–280, Apr. 2004, doi: 10.1080/03081070310001633581.

[6] L. Chandi, C. Silva, T. Gualotuña, and D. Martinez, "Model for selecting software development methodology," in *Proc. Int. Conf. Inf. Technol. Syst. (ICITS)*, vol. 721, Á. Rocha T. Guarda, Eds., Springer, 2018, pp. 62–73, doi: 10.1007/978-3-319-73450-7_7.

[7] P. Clarke and R. V. O'Connor, "The situational factors that affect the software development process: Towards a comprehensive reference framework," *Inf. Softw. Technol.*, vol. 54, no. 5, pp. 433–447, May 2012, doi: 10.1016/j.infsof.2011.12.003.

[8] G. Coleman and R. V. O'Connor, "An investigation into software development process formation in software start-ups," *J. Enterprise Inf. Manage.*, vol. 21, no. 6, pp. 633–648, Oct. 2008, doi: 10.1108/17410390810911221.

[9] B. Efe, "An integrated fuzzy multi criteria group decision making approach for ERP system selection," *Appl. Soft Comput.*, vol. 38, pp. 106–117, Jan. 2016, doi: 10.1016/j.asoc.2015.09.037.

[10] S. Faraj and L. Sproull, "Coordinating expertise in software development teams," *Manage. Sci.*, vol. 46, no. 12, pp. 1554–1568, Dec. 2000, doi: 10.1287/mnsc.46.12.1554.12072.

[11] D. Gaur and S. Aggarwal, "Selection of software development model using TOPSIS methodology," in *Data and Communication Networks* (Advances in Intelligent Systems and Computing), vol. 847, L. C. Jain, V. E. Balas, and P. Johri, Eds., Berlin, Germany: Springer, 2019, pp. 123–133, doi: 10.1007/978-981-13-2254-9_11.

[12] N. Govil and A. Sharma, "Validation of agile methodology as ideal software development process using fuzzy-TOPSIS method," *Adv. Eng. Softw.*, vol. 168, Jun. 2022, Art. no. 103125, doi: 10.1016/j.advengsoft.2022.103125.

[13] V. Guntamukkala, H. J. Wen, and J. M. Tarn, "An empirical study of selecting software development life cycle models," *Human Syst. Manage.*, vol. 25, no. 4, pp. 265–278, Nov. 2006, doi: 10.3233/hsm-2006-25405.

[14] Y. A. Harb, C. Noteboom, and S. Sarnikar, "Evaluating project characteristics for selecting the best-fit agile software development methodology: A teaching case," *J. Midwest Assoc. Inf. Syst. (JMWAIS)*, vol. 1, no. 1, p. 4, 2015.

[15] C. Hestomo, E. K. Budiardjo, and A. Ferdinansyah, "Quality function deployment analysis in selecting software development methodology: Case study of ABC-CORP," in *Proc. 2nd Int. Conf. Softw. Eng. Inf. Manage.*, Jan. 2019, pp. 63–68, doi: 10.1145/3305160.3305202.

[16] M. Hicdurmaz, "A fuzzy multi criteria decision making approach to software life cycle model selection," in *Proc. 38th Euromicro Conf. Softw. Eng. Adv. Appl.*, Sep. 2012, pp. 384–391, doi: 10.1109/SEAA.2012.71.

[17] A. K. M. Z. Islam and D. A. Ferworn, "A comparison between agile and traditional software development methodologies," *Global J. Comput. Sci. Technol.*, pp. 7–42, Dec. 2020, doi: 10.34257/gjcstcvol20is2pg7.

[18] P. M. Khan and M. M. S. S. Beg, "Extended decision support matrix for selection of SDLC-models on traditional and agile software development projects," in *Proc. 3rd Int. Conf. Adv. Comput. Commun. Technol. (ACCT)*, Apr. 2013, pp. 8–15, doi: 10.1109/ACCT.2013.12.

[19] S. S. Kute and S. D. Thorat, "A review on various software development life cycle (SDLC) models," *J. Res. Comput. Commun. Technol.*, vol. 3, no. 7, pp. 778–779, 2014.

[20] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical study of agile software development methodologies: A comparative analysis," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 1, pp. 1–6, Feb. 2015, doi: 10.1145/2693208.2693233.

[21] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *J. Syst. Softw.*, vol. 82, no. 11, pp. 1869–1890, Nov. 2009, doi: 10.1016/j.jss.2009.05.052.

[22] M. I. Hossain, "Software development life cycle (SDLC) methodologies for information systems project management," *Int. J. Multidisciplinary Res.*, vol. 5, no. 5, p. 6223, Sep. 2023, doi: 10.36948/ijfmr.2023.v05i05.6223.

[23] F. O. Albalawi and M. S. Maashi, "Selection and optimization of software development life cycles using a genetic algorithm," *Intell. Autom. Soft Comput.*, vol. 28, no. 1, pp. 39–52, 2021, doi: 10.32604/iasc.2021.015657.

[24] V. Rastogi, "Software development life cycle models-comparison, consequences," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 1, pp. 168–172, 2015.

[25] R. A. Ribeiro, A. M. Moreira, P. van den Broek, and A. Pimentel, "Hybrid assessment method for software engineering decisions," *Decis. Support Syst.*, vol. 51, no. 1, pp. 208–219, Apr. 2011, doi: 10.1016/j.dss.2010.12.009.

[26] J. Sheffield and J. Lemétayer, "Factors associated with the software development agility of successful projects," *Int. J. Project Manage.*, vol. 31, no. 3, pp. 459–472, Apr. 2013, doi: 10.1016/j.ijproman.2012.09.011.

[27] S. Soobia. et. al., "Analysis of software development methodologies," *Int. J. Comput. Digit. Syst.*, vol. 8, no. 5, pp. 445–460, Jan. 2019, doi: 10.12785/ijcds/080502.

[28] A. Srivastava, B. Singh, A. Chabra, and R. Majumdar, "Application and use of MCDM technique in software industry," in *Proc. Int. Conf. INFOCOM Technol. Unmanned Syst.*, Dec. 2017, pp. 487–491, doi: 10.1109/ICTUS.2017.8286058.

[29] N. P. Suh, "Axiomatic design theory for systems," *Res. Eng. Des.*, vol. 10, no. 4, pp. 189–209, Dec. 1998.
[30] L. R. Vijayasarathy and C. W. Butler, "Choice of software development methodologies: Do organizational, project, and team characteristics matter?" *IEEE Softw.*, vol. 33, no. 5, pp. 86–94, Sep. 2016, doi: 10.1109/MS.2015.26.
[31] M. R. Wrobel, "Emotions in the software development process," in *Proc. 6th Int. Conf. Human Syst. Interact. (HSI)*, Jun. 2013, pp. 518–523, doi: 10.1109/HSI.2013.6577875.

TANER BEHÇET KEPKEP received the degree from the Department of Computer Engineering, Atılım University, in 2006, and the master's degree in computer science from Oxford Brookes University, U.K., in 2009. He is currently pursuing the Ph.D. degree. He is a Freelance Software Engineer. He has work experience in various provinces of Türkiye and abroad.

ALPTEKİN DURMUŞOĞLU received the B.Sc. degree in industrial engineering and management (double major program) from Çankaya University, in 2004, and the M.Sc. degree in industrial engineering from Gaziantep University, in 2008. He is currently with the Industrial Engineering Department, Samsun University, Türkiye. His research interests include techno-investments, technology management, and management information systems.

TÜRKAY DERELİ is currently with the Office of President, Hasan Kalyoncu University, Gaziantep, Türkiye. He has published numerous technical papers in professional academic journals and conferences and has several textbooks on CAD/CAM, ICT, and quality management. He is also an active referee for many professional journals and edited several conference proceedings. His current research interests include innovation and technology management, CAD/CAM, process planning, feature technology, TQM, agile and responsive manufacturing and management, soft computing, and informatics and applications of artificial intelligence.