IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 23, 2017, accepted January 26, 2018, date of publication February 13, 2018, date of current version March 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2805702

Statistical Analysis of the Effects of Heavyweight and Lightweight Methodologies on the Six-Pointed Star Model

MUHAMMAD AZEEM AKBAR[1,2], JUN SANG[1,2], ARIF ALI KHAN[3], (Senior Member, IEEE), FAZAL-E-AMIN[4], NASRULLAH[1,2], SHAHID HUSSAIN[3], MOHAMMAD KHALID SOHAIL[3], HONG XIANG[1,2] AND BIN CAI[1,2]

[1]Key Laboratory of Dependable Service Computing in Cyber Physical Society of Ministry of Education, Chongqing University, Chongqing 400044, China
[2]School of Software Engineering, Chongqing University, Chongqing 401331, China
[3]COMSATS Institute of Information Technology, Islamabad 54700, Pakistan
[4]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia

Corresponding author: Jun Sang (jsang@cqu.edu.cn)

This work was supported by the National Key R&D Program of China under Grant 2017YFB0802400.

ABSTRACT Traditionally, software development organizations relied on heavyweight development methodologies, such as waterfall, V-model, and others. Later, agile development methodologies known as lightweight methodologies were introduced. Many considered these to be more flexible and more effective than heavyweight methodologies. Both methodologies are equally important for a software development life cycle. The purpose of adopting software development methodologies is to optimize the process model to achieve milestones while concurrently and effectively managing time, budget, and quality. The literature review reveals that there is a lack of statistical evidence for determining the effect of both methodologies on the six-pointed star model (schedule, scope, budget, risk, resource, and quality). In this paper, statistical comparisons were performed for the effects of both methodologies on each factor of the six-pointed star model and the interdependency among factors. Numerical analyses were conducted based on survey responses collected from the experienced users of both methodologies. After examining the results of all the factors of both methodologies, it was determined that lightweight methodologies are suitable for small-scale projects and that heavyweight methodologies perform better for medium- and large-scale projects.

INDEX TERMS Heavyweight methodologies, lightweight methodologies, software development life cycle, software requirement specification.

I. INTRODUCTION

Software systems have a great impact on everyday life. In the beginning of the software development era, the development activities were performed informally by following the messy software development activities from requirement gathering to the maintenance phase. This procedure was considered to be effective for small projects. Software engineering has become a growing and emerging field due to the significance of software systems in every aspect of life. As software engineering continued its growth, its projects became more complex. Therefore, the development of formal software development methodologies became vital for achieving efficient and highly effective software systems.

In the early 1990s, due to the boom in software and hardware industries, it was realized that project management methodologies are helpful for achieving excellent results in the production of both software and hardware products. Due to the adoption of software development methodologies, organizations became more efficient when producing high-quality products within a specified time and budget [1], [2]. It is the priority of every organization to choose low-cost software development methodologies that could provide best practices to develop high-quality products and fulfill the organization requirements [3]. Therefore, software development methodologies are considered to be key factors for the success and progression of the organizations. Many methodologies have been introduced for software projects, which are broadly categorized as heavyweight and lightweight software development methodologies [5].

The initially proposed methodologies were plan-driven and document-oriented. In the plan-driven methodologies, the set of complete software requirements specification (SRS) is constructed and followed by high-level design, architectural activities and inspection activities [6]. The plan-driven methodologies were considered heavyweight methodologies. Heavyweight methodologies have low change rates [7]–[9]. Several experts have developed their own methodologies and applied them in order to assess expected outcomes. These methodologies were iterative and incremental. They were introduced in 1975 and eventually became the bases for the modern agile methods [7]–[10]. In 2001, a meeting was held in which 17 process methodologists and project managers participated. The motivation of the meeting was to highlight the future trends of software development. They combined different methodologies and decided to name these methodologies as "Agile", meaning sufficient and light [11]. That meeting was declared the father of agile process methods. They are commonly known as lightweight software development methodologies and are now considered to be popular development techniques [2], [7].

Software development methodologies are equally important for both organizations and customers [3], [12]. The software development methodologies are broadly categorized as lightweight and heavyweight methodologies. Both methodologies have their own importance. When considering development methodologies, all factors of the project should be taken into consideration [13]. Hence, the key objective of this work is to highlight the best software development methodology based on the size of the project.

The remainder of the paper is organized as follows. Section II describes the literature, characteristics and comparison of heavyweight and lightweight methodologies. In Section III, the outlines of the research methodology are provided. Section IV describes the data collection techniques and respondents' information. Section V provides the results and discussions. Lastly, Section VI is where the conclusions are given.

II. BACKGROUND STUDY

A. CHARACTERISTICS OF HEAVYWEIGHT METHODOLOGIES

The traditional methodologies are considered heavyweight methodologies in which the working process is a sequential series from outlining requirements to deployment [14], [15]. Heavyweight methodologies impose an orderly process to make the development process more efficient and predictable. Heavyweight methodologies are document-oriented (commonly known as SRS) and are used throughout the software development life cycle [16], [17]. At the end of the 1950s, many methodologies were applied to software project management. The primary methodologies were used to discover the best way of collecting requirements, dealing with problems and developing the solutions for the discovered problems in systematic ways. Some methodologies were iterative and some were incremental in nature [10]. The other types of methodologies were linear or sequential and are called the Waterfall Model [18]. In the waterfall model, it is assumed that the engineering team has perfect knowledge about the project's requirements at the initial phase. Therefore, on the basis of the initial gathered requirements, all SDLC steps are well planned. All steps are followed in a linear sequential fashion in order to achieve the goal. Changing requirements is discouraged due to the negative effect on the budget and time [19]. The V-Model is yet another software development model, which starts from the user's requirements and finishes at the finalized development of the system [20]. This model has many constrictions on testing and verification. Every phase of the model must be tested and verified. Development begins with low-level components and is completed with high-level components before the entire system is verified. The iterative model was introduced to develop and deliver the project in iterations instead of delivering the entire project at once [21]. The project is sliced into appropriate iterations. At the start of every iteration, the obligatory requirements for the imminent iteration are collected. Every delivered iteration is an addition to the previously delivered system [10].

Heavyweight methodologies are popular due to the sequential process and its success under certain circumstances. There are some criticisms of heavyweight methodologies due to their bureaucratic approach in which the pace of project development is slow [22].

1) PREDICTIVE APPROACH

Heavyweight software development methodology tends to be a long process due to the scope of the detailed planning at the beginning. It is followed by an engineering discipline approach where the development is predictive and repeatable. It emphasizes drawing the system requirements and the effectiveness and efficiency with which these requirements are met.

2) COMPREHENSIVE DOCUMENTATION

In heavyweight software development methodologies, SRS is the system that is considered the main part of the documentation. The key tasks in traditional methodologies are the big upfront design processes. Therefore, it is a must to collect all the customers' expectations and the true requirements and to have the customer sign off prior to writing the code. Hence, this approach has proved to be very successful in the engineering disciplines, which makes it attractive to the software development industry.

3) PROCESS-ORIENTED

The key idea of heavyweight methodologies is to determine the process that will work best and implement it to achieve the milestone [22]. The determining process must have some tasks that are performed by the project managers, software designers, software coders and testing teams. A well-defined procedure is used to conduct and operate these tasks.

4) TOOL-ORIENTED

The heavyweight methodologies are tool-oriented for project management, code editors, compilers, and others.

They should be used for the successful completion of the project.

B. CHARACTERISTICS OF LIGHTWEIGHT METHODOLOGIES

Due to rigidness, heavy dependency on documentation and long-term planning from the initial stage, the traditional methodologies are not very attractive for software development organizations [23]. Instead, Agile methodologies are preferred due to having a people–oriented approach, smaller planning phases, very light documentation, and quicker accepted changes. Many agile methodologies exist. Every agile methodology has the target to develop a friendly, understandable system and rapidly provide requirements in a changing environment [24]. A frequently developed release methodology was introduced by Extreme Programming (XP) in which developers work in pairs for continuous code review. This methodology emphasizes test-driven development and produces very robust, high-quality software. Each iteration cycle of XP includes design, bug fixing and refactoring [25]. Scrum is a well-known agile methodology based on the principles of lean manufacturing [23]. In scrum, a project is planned as a short-term project (usually for 7 to 30 days) and the requirements are collected and focused by conducting frequent, short meetings. The scrum board is used for tracking the tasks. Hence, there is a scrum master tasked with imposing the rules and shielding the team from distractions [26], [27]. Rapid Application Development (RAD) is the simplest adopted agile methodology. This methodology emphasizes minimizing the planning, focusing the prototyping and reusing the components. This methodology proved very effective where the prototyping was enough to serve as the final product, but experienced team members are required [28], [29].

Highsmith and Cockburn [30] stated that the agile methodologies are not the process. The process is the recognition of team members acting as the key drivers for the success of the project. The selected team members act with intense focus on maneuverability and effectiveness. This yield provides new principle values that describe a vision of an agile world. The characteristics of lightweight methodologies are as found below.

1) PEOPLE-ORIENTED

Lightweight methodologies are considered people- or customer-oriented. The important success factors in lightweight methodologies are developers, stakeholders and end users. The project success rate depends on the agile team members [30]. According to Highsmith and Cockburns [31], the most significant impact of the managers is that people factors are more emphasized, including sociability, skill, talent and communication.

2) ADAPTIVE

Lightweight methodologies permit changes to requirements and the status of the project [32]. Today, it is not a challenge to stop changes. The concentrated focus is on how the changes are handled in an effective and efficient way. The externally occurring changes that result from environmental causes are a critical variation since these types of changes are not reducible, especially since decreasing the budget is the typical response [30].

3) CONFORMANCE TO ACTUAL

The lightweight methodologies are opposed to detailed plans and instead conformance is treated as the actual values. According to Highsmith, agility is controlled as conformance to the business values instead of control within the conformance plan [33]. The end user decides the business values that will be added at every iteration of the development cycle to the ongoing product.

4) BALANCING FLEXIBILITY AND PLANNING

Planning is very important, but in lightweight methodologies, the future prediction of the project is challenging. Due to the consideration of multiple variables, the best way to plan is a weekly detailed plan, a rough plan for the coming month and a very crude plan for beyond [11]. Therefore, the vision is that, while making decisions, the likelihood of variability should be taken into account at every stage [29].

5) EMPIRICAL PROCESS

When using lightweight methodologies, software is developed in an empirical way. In engineering, methods are either empirical or defined. Therefore, the defined processes are not considered due to the demanding changes and the adoption of required changes during development. According to Laurie Williams, defined processes are unlikely due to the non-acceptance of required changes since, when requirements change, technology will also be changed [30].

6) DECENTRALIZED APPROACH

Lightweight methodologies that use a decentralized management style have a positive impact on a software project rather than an autocratic management process. In lightweight methodologies, decisions are spread out to the developers, but this does not mean that the developers take part in management. Instead, developers support the management team when they are making appropriate technical decisions [7].

7) SIMPLICITY

The heavyweight methodology team always takes the simplest way necessary to achieve its goals. Project teams do not think about tomorrow's problems [32]. Due to the simplicity, it is easy to manage the design changes, never produce more than what is required and always try to develop the appropriate project [34].

8) COLLABORATION

The customer is closely involved throughout the SDLC. This has proven helpful for the development team in receiving regular feedback from the customer [32]. The decentralized approach of the lightweight methodologies encourages collaborative discussions.

9) SMALL SELF-ORGANIZED TEAMS

The teams in lightweight methodologies are self-organized. All aspects of the project are communicated to the team and the team chooses the best way to achieve them. Lightweight methodologies are effective in adopting small development teams. However, a large team is difficult to manage and the ideas used to achieve the milestone are dispersed [31].

C. BRIEF COMPARISON OF LIGHTWEIGHT AND HEAVYWEIGHT METHODOLOGIES

Traditional software development methodologies were introduced a long time ago. In 1970, Royce presented a sequential methodology for software development named the waterfall model, which has proven very useful [38]. Despite the project success, it has had many limitations, such as linearity and inflexible requirements a formal process of development with respect to the project size. Hence, the drawbacks were addressed by Kent Beck when he launched Extreme programming [39]. This became the first agile methodology and proved very effective in software development. The agile methodology dealt with volatile and unstable requirements by supporting a number of techniques and focusing the collaboration among customers and developers that support early product deployment. A brief comparison of both methodologies is illustrated in Table 1.

TABLE 1. Comparison of heavyweight and lightweight methodologies.

Description	Lightweight methodology	Heavyweight methodology
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to Change	Adaptability	Sustainability
Culture	Leadership Collaboration	Command-and-Control
Documentation	Low	Heavy
Emphasis	People Oriented	Process Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in Project	End of Project
Team Size	Small/Creative	Large

III. RESEARCH METHODOLOGY

Both methodologies have similarities and differences, as previously discussed. The existing analyses are not based on the numerical data but rather depend upon the user's implementation, experience and opinions about both methodologies. To the contrary, our analysis is based on the six-pointed star model of project management (Fig. 2) rather than traditional project input/output and process factors. Traditionally, the considered factors for successful software development are time, costs and scope [35]. The project management body of knowledge (PMBOK 4.0) provides an updated triple constraints model based on six factors that include schedule, scope, budget, risk, resources, and quality. Every factor has its own importance for the success of the software project [36], [37]. The presented six-pointed star model is divided into two triangles in which the first one contains input/output factors of schedule, scope, and budget (Fig. 3), while the other triangle contains risk, resources and quality (Fig. 4).

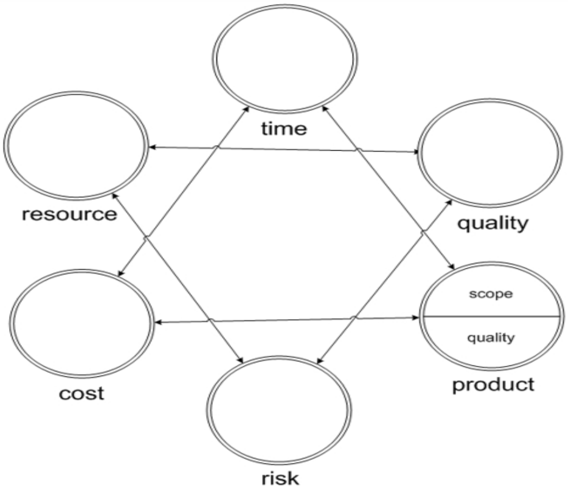


FIGURE 1. Six-pointed star model (wikipedia.org/wiki/File: TripleConstraint.jpg.)

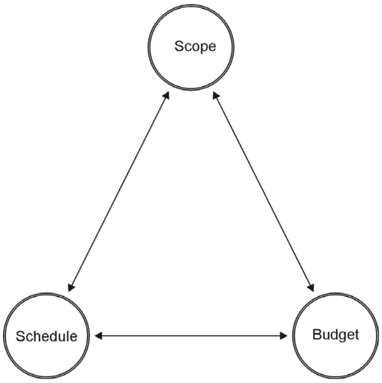


FIGURE 2. First triangle of the six-pointed star.

The schedule factor is responsible for the on-time completion of the project. Scope focuses on the goals and mission of the project and the requirements. Budget is responsible for meeting the requirements of the project within the declared budget and achieving the targeted return on investment. Risk identifies the causes of risk and manages them. Resources are responsible for the assurance of the availability

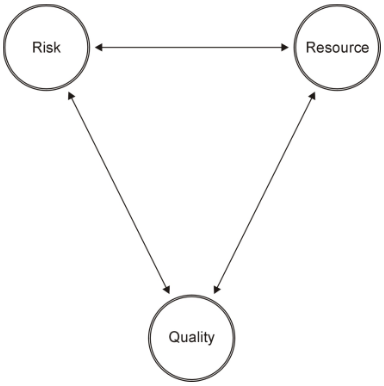


FIGURE 3. Second triangle of the six-pointed star.

of both the personnel and material needed for the project. Quality is related to the inclusive success of the project. Hence, we check the statistical effects of both the methodologies on each factor of the six-pointed star model.

IV. DATA COLLECTION

A survey was used to collect the numerical responses from the subjects. For this purpose, the developed questionnaire was categorized into three sections. The first section contains the general information about the respondent and their organization. The second section contains questions related to the factors of the six-pointed star model (as shown in Table 1). The third section contains queries to check the relationship of one factor of the six-pointed star to other factors. The survey was conducted from January to April (2017) and twenty organizations participated in this survey. To assess the survey questions, a five-point Likert scale was used (strongly disagree, disagree, neutral, agree and strongly agree). The collected data was summarized and statistical graphical techniques were applied by using well known statistical tools.

According to general information, approximately 57% of the respondents belong to a software organization, 18% belong to an educational organization (Doctoral student), 13% belong to telecommunications, and the remaining 12% are from other fields. Furthermore, 43% of the organizations have a range of 50 – 100 staff members, 37% have a range of 20 – 50 staff members, and 20% have a range of 100 – 300 staff members. Approximately 52% of the respondents used capability quality standards. According to the respondents, 70% adopted new methodologies\ technologies from the market leader. For lightweight methodologies, 60% of the respondents stated that they have average knowledge and 23% of the respondents stated that they have extensive knowledge of the methodologies. For knowledge of heavyweight methodologies, 52% of the respondents stated that they have very extensive knowledge, 43% of the respondents stated that they have extensive knowledge and the remaining 5% have average knowledge. Additionally, 52% of the participants responded that the most appealing element of lightweight methodology was people orientation versus process orientation and 36% stated that it was working with code versus documentation. According to general views of respondents, lightweight methodologies provide better results for small- and medium-scale projects and heavyweight methodologies provide better results for large-scale projects.

TABLE 2. Queries related to the six-pointed star model for both methodologies.

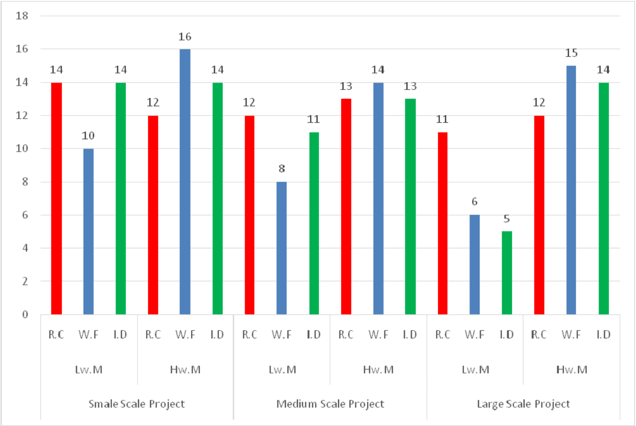
Factor	Survey questions	Q. No.
Scheduling	Do the project teams get the satisfactory requirements from the customer by using both methodologies?	2.1
	Is the project delivered on time according to schedule by using both methodologies?	2.2
	Is the workflow followed as described in the scheduling by using both methodologies?	2.3
Scope	Does the project have a well-defined scope by using both methodologies?	3.1
	Do team members have a very clear scope of the project by using both methodologies?	3.2
Budget	Is the project completed within the budget by using both methodologies?	4.1
	Does the project provide a good Return on Investment by using both methodologies?	4.2
Risk	Are the project risk and opportunities managed in both methodologies?	5.1
	Are business objectives met by using both methodologies?	5.2
Resources	Are human and material resources mostly available in both methodologies?	6.1
	Is the maximum utilization of available resources achieved in both methodologies?	6.2
Quality	Are quality requirements met by using the lightweight methodology?	7.1
	Is client satisfaction met by using the lightweight methodology?	7.2
	Is the project successful by using the lightweight methodology?	7.3

V. RESULTS AND DISCUSSIONS

According to the second part of the survey questionnaire, wide data were collected and results were finalized. The Likert scale was categorized into three phases. The phases are as follows: a) strongly agree + agree, b) neutral, and c) strongly disagree + disagree. For presenting the results, only part "a" will be used, which shows the respondents' opinions about each factor. For more specific opinions, projects are categorized as small-scale, medium-scale and large-scale projects.

A. SCHEDULE FACTOR

The results of the schedule factors are presented in figure 4. It shows that in small-scale projects, collecting requirements is more significant in lightweight than in heavyweight methodologies. It also shows that in-time delivery is equal in both methodologies, but work flow is more significant in heavyweight than in lightweight methodologies. Furthermore, by determining the results of medium- and large-scale projects, it is shown that increasing the project size decreases the significance of the lightweight methodology. Therefore, lightweight methodologies are more effective for small-scale projects, and the schedule factor is more effective for heavyweight methodologies.



R.C= Requirement Completed, W.F= Work Flow, I.D= In-time Delivery

FIGURE 4. Effect of heavyweight and lightweight methodologies on the schedule factor.

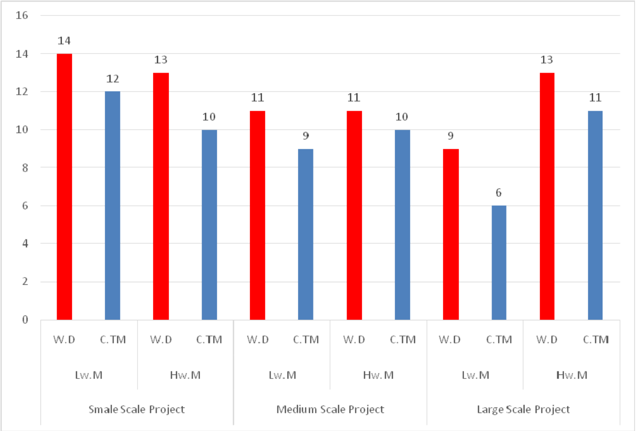


FIGURE 5. Effect of heavyweight and lightweight methodologies on the scope factor.

B. SCOPE FACTOR

Figure 5 shows a graphical representation of the scope factor. By using lightweight methodologies for small-scale projects, the project scope is well defined and clearer compared to the heavyweight methodologies. For medium scale projects, both methodologies are equally efficient for defining and understanding the project scope. Furthermore, for large-scale projects, heavyweight methodologies are much better for both variables than lightweight methodologies.

C. BUDGET FACTOR

Figure 6 illustrates the results for the budget factor. Lightweight methodologies lead to more satisfactory project completions within budget and returns on investment in small-scale projects as opposed to using heavyweight methodologies. For medium-scale projects, the return on investment is the same in both methodologies, but the completion within budget is better using heavyweight methodologies rather than lightweight. For large-scale projects, heavyweight methodologies are much more significant in both queries as opposed to lightweight methodologies.

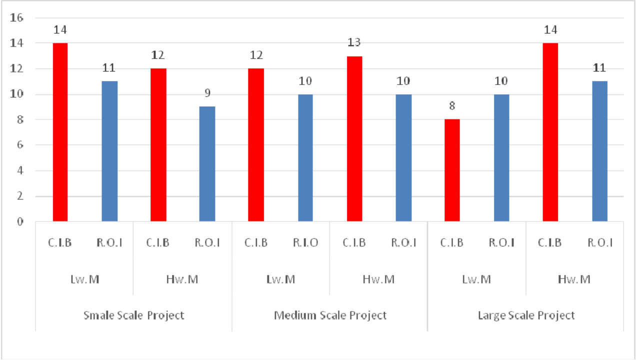


FIGURE 6. Effect of heavyweight and lightweight methodologies on the budget factor.

Hence, the graph shows that the budget factor in lightweight methodologies is the best for small- and medium-scale projects, but heavyweight methodologies are more satisfactory in large-scale projects.

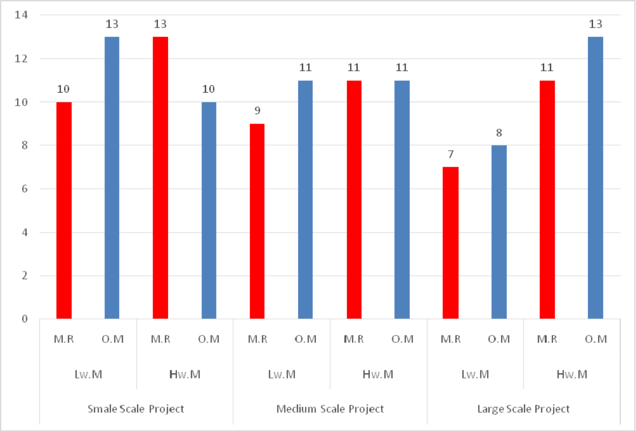


FIGURE 7. Effect of heavyweight and lightweight methodologies on the risk factor.

D. RISK FACTOR

The risk factor is key in software project management. Figure 7 provides a graphical representation that shows the results of the risk factor. Lightweight methodologies are poorer than heavyweight methodologies for managing project risk and opportunities for all scales of projects. Successfully meeting the objectives of the business requires lightweight methodologies to be superior in small-scale projects, whereas both methodologies are equal in medium-scale projects. For large-scale projects, heavyweight methodologies are much more significant than lightweight methodologies.

E. RESOURCE FACTOR

Figure 8 shows the graphical representation of the concluded results of resource factor. Lightweight methodologies are of greater significance for the availability and better utilization

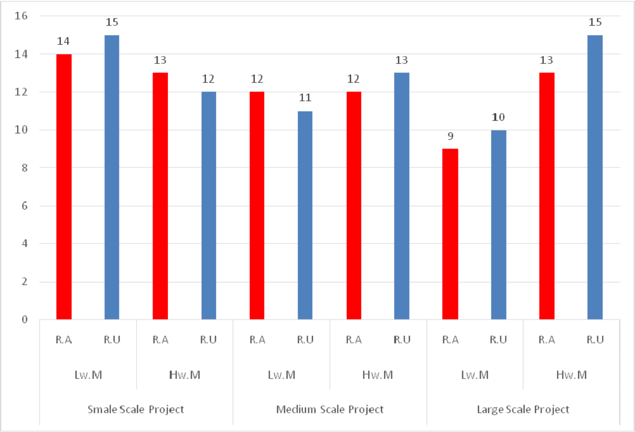


FIGURE 8. Effect of heavyweight and lightweight methodologies on the resource factor.

of resources than heavyweight methodologies in small-scale projects. In medium-scale projects, resource availability is the same in both methodologies but the utilization of resources is the best in heavyweight methodologies. For large-scale projects, the availability and utilization of resources have better results using heavyweight methodologies compared to the lightweight methodologies.

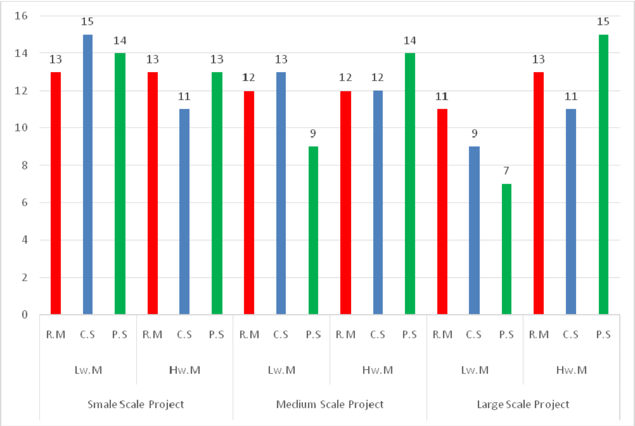


FIGURE 9. Effect of heavyweight and lightweight methodologies on the quality factor.

F. QUALITY FACTOR

Figure 9 illustrates the graphical representation of the quality factor. Lightweight methodologies are of greater significance in small-scale projects rather than heavyweight methodologies. Lightweight methodologies have greater client satisfaction and project success scores and an equivalent quality requirement score as heavyweight methodologies. For medium-scale projects, quality requirements are equal in both methodologies, whereas client satisfaction is better with lightweight methodologies. However, the project success rate is much better with heavyweight rather than lightweight methodologies. For the development of large-scale projects, heavyweight methodologies are appreciably better in all quality factor variables

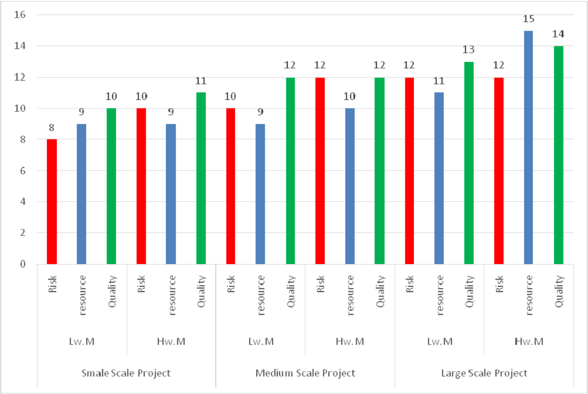


FIGURE 10. Effect on risk, resource and quality factors with respect to budget.

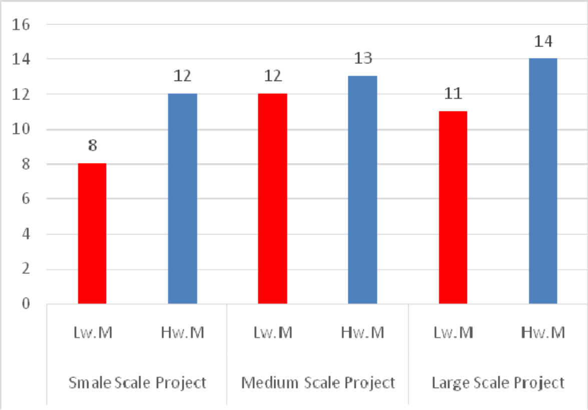


FIGURE 11. Effect on quality factor with respect to schedule factor.

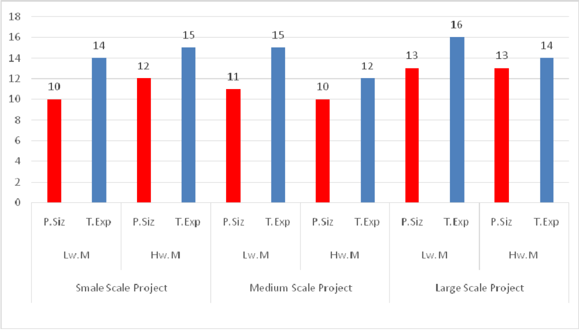


FIGURE 12. Effect on quality factor with respect to project size and experience of the team.

G. EFFECT ON RISK, RESOURCE AND QUALITY FACTORS WITH RESPECT TO BUDGET

Budget is a sensitive factor. Therefore, the budget affects the different factors of the six-pointed star model differently. The graphical representation in figure 10 presents the effects of the budget factor on risk, resource and quality factors under both methodologies. For small-scale projects, heavyweight methodologies are more effective than lightweight methodologies for the three variables. For medium-scale projects, the effects on quality factors are the same in both methodologies, but risk and resource factors are more affected in heavyweight methodologies due to the changing budget.

Section- A1 (Respondent Information)		
Full Name (optional)		Job Title Position
Have you ever been participated in an software development methodology	Yes □	No □
Working Experience (Years) in Software development organization		
What is the scope of your company?	Client □	Vendor □
	Not sure □	Other
Email Address		
Current address of your organization including country		
How many years of industry/ academia experience do you have in your field?		
Have you ever participated in Software development life cycle Improvement Project?	Yes □	No □
Section- A2 (Organization Detail)		
Name of Organization (Optional)		
What is the primary business function of your organization? (You may tick more than one)	Collocated Software development □	Global/offshore Software development □
	Research □	Other
Please specify the size of your organization.	Small □	Medium □
	Large □	Not sure □
Please specify the number of employees in your	Less than 50 □	51-100 □
	101-150 □	Greater than 150 □
Please specify the type of your organization	National □	Multinational □
	Not sure □	Other
Does your organization adopted Software development life cycle Process Improvement standards or (CMMI/ISO)	CMMILevel-1 □ (Initial)	CMMILevel-2 (Managed) □
	CMMILevel-3 □ (Defined)	CMMILevel-4(Quantitatively □ Managed)
	CMMILevel-5 □ (Optimizing)	ISO □
	Other	Not sure □
Which software development methodology your organization adopted		
Which factor is most important for you as a professional when adopting a method	Low cost □	Easy to handle □
	Great productivity □	Great reliability □
	Other	Not sure □

Section: B1 –Questions related to six pointed star model.

The aim of this section is to collect factor wise responses from the participants which make it possible to analyze the proposed work according to six pointed star model provided by the project management Body of knowledge (PMBOK 4.0). The factor wise collection of data make us enable to check that which methodology is efficient for what purpose to what extent.

Please rank each question according to your own understanding and experience about software development methodologies.

SSP= Small Scale Project, MSP=Medium Scale Project, LSP=Large Scale Project, SDA= Strong Disagree, DA= Disagree, N= Neutral, A=Agree, S A=Strong agree

Factors	**Questions**	**SSP**	**MSP**	**LSP**
Schedule Factor	Project teams gets the satisfactory requirements from the customer by using lightweight methodology.	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □

	Project teams gets the satisfactory requirements from the customer by using heavyweight methodology.	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Project is delivered on time according to schedule by using lightweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Project is delivered on time according to schedule by using heavyweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Working flow is followed accordingly, described scheduling in lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Working flow is followed accordingly, described scheduling in heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Scope factor	Project usually has a well-defined scope by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Project usually has a well-defined scope by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Team members have very clear scope of the project by sing lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Team members have very clear scope of the project by sing heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Budget factor	Project completed within budget by using lightweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Project completed within budget by using heavyweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The project provides good Return on Investment by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The project provides good Return on Investment by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □

Risk factor	Project risk and opportunities are managed in lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Project risk and opportunities are managed in heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Business objectives are meet by using lightweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Business objectives are meet by using heavyweight methodology?	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Resource factor	Human and material resources are mostly available in lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Human and material resources are mostly available in heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Maximum available resources are utilized in lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Maximum available resources are utilized in heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Quality factor	Quality requirements are met by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Quality requirements are met by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Client satisfaction is met by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	Client satisfaction is met by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The project is successful overall by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □

	The project is successful overall by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Section: B2 –Questions related to the dependency of one factor of six pointed star model on other factor. The aim of this section is to check the dependency of one factor on other factor. Please rank each question according to your own understanding and experience about software development methodologies.				
Dependency of risk on budget	The Risk factor is effected due to change in budget in lightweight methodologies	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The Risk factor is effected due to change in budget in lightweight methodologies	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Dependency of resource factor on budget	The Resource factor is effected due to change in budget by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The Resource factor is effected due to change in budget by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Dependency of quality factor on budget factor	The quality factor is effected due to change in budget by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The quality factor is effected due to change in budget by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Dependency of quality factor on schedule factor	The quality factor is effected due to change in schedule factor by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The quality factor is effected due to change in schedule factor by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Dependency of quality factor on project size and team experience	The quality factor is effected due to project size and experience of team members by using lightweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
	The quality factor is effected due to project size and experience of team members by using heavyweight methodology	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □	SDA □ DA □ N □ A □ SA □
Section: C- Add your comment if any				

For large-scale projects, the effects on the risk factor are the same in both methodologies, but resource and quality factors are more effective in heavyweight methodologies rather than lightweight methodologies. Therefore, it is determined that heavyweight methodologies are more sensitive than lightweight methodologies with respect to the budget factor.

H. EFFECT ON THE QUALITY FACTOR WITH RESPECT TO THE SCHEDULE FACTOR

Figure 11 shows the graphical representation of the effect on quality factors with respect to the schedule factor. Heavyweight methodologies are more affected by schedule changes. Therefore, the analyzed results show that heavyweight methodologies are very rigid. The schedule factor plays a significant role in the success of the project using heavyweight methodologies.

I. EFFECT ON THE QUALITY FACTOR WITH RESPECT TO PROJECT SIZE AND THE EXPERIENCE OF THE TEAM

According to the summarized results shown in figure 12, project size and the expertise of the project team members affect the lightweight methodologies more than the heavyweight methodologies for all types of projects. Hence, the quality of the projects using lightweight methodologies highly depends on the project size and the expertise of the team members.

VI. CONCLUSIONS

The analyses were conducted in order to define the characteristics of the software development methodologies and determine the best methodology according to the project and organizational requirements. According to the summarized results in the factors of the six-pointed star model, it is proven that almost all factors favor lightweight methodologies for small-scale projects. When testing medium-scale projects, there is a huge fluctuation between the variables of all factors for both methodologies, but it appears that both methodologies are quite similar in this case. Nonetheless, heavyweight methodologies are better in some measures to some extent. The concluded results of both methodologies for large-scale projects show that heavyweight methodologies are much more satisfactory for all factors of the six-pointed star model. When examining the effect of the budget factor on risk, resource and quality factors, heavyweight methodologies are much more sensitive than lightweight methodologies for all sized projects. When analyzing the effect of the schedule factor on the quality factor, it is determined that heavyweight methodologies are largely affected in all scale projects. While examining the effect of project size and developer expertise, it is determined that lightweight methodologies are much more sensitive.

Therefore, it is concluded that quality of the product and success rate depends upon team size and developer experience by adopting lightweight methodologies. In conclusion, after examining the results of all factors upon both methodologies, it is determined that lightweight methodologies are the best for small-scale projects and that heavyweight methodologies are better for medium- and large-scale projects.

In the future, we plan to analyze the effect of software development methodologies on the six-pointed star model in global software development.

APPENDIX

See Questionnaire.

REFERENCES

[1] 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./Oct. 2016.

[2] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," *Robot. Comput.-Integr. Manuf.*, vol. 43, pp. 59–67, Feb. 2015.

[3] M. Niazi *et al.*, "Challenges of project management in global software development: A client-vendor analysis," *Inf. Softw. Technol.*, vol. 80, pp. 1–19, Dec. 2016.

[4] N. M. A. Munassar and A. Govardhan, "A comparison between five models of software engineering," *Int. J. Comput. Sci. Issues*, vol. 7, no. 5, pp. 95–101, 2010.

[5] A. I. Khan, R. J. Qurashi, and U. A. Khan. (2011). "A comprehensive study of commonly practiced heavy and light weight software methodologies." [Online]. Available: https://arxiv.org/abs/1111.3001

[6] A. Aitken and V. Ilango, "A comparative analysis of traditional software engineering and agile software development," in *Proc. 46th Hawaii Int. Conf. IEEE Syst. Sci. (HICSS)*, Jan. 2013, pp. 4751–4760.

[7] M. A. Awad, "A comparison between agile and traditional software development methodologies," School Comput. Sci. Softw. Eng., Univ. Western Australia, Perth, WA, Australia, Tech. Rep., 2005, pp. 1–38.

[8] A. Gunasekaran, "Agile manufacturing: A framework for research and development," *Int. J. Prod. Econ.*, vol. 62, nos. 1–2, pp. 87–105, 1999.

[9] R. T. Futrell, D. F. Shafer, and L. Shafer, *Selecting Software Development Life Cycles, Quality Software Project Management*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002, pp. 101–161.

[10] C. Larman and V. R. Basili, "Iterative and incremental developments: A brief history," *IEEE Comput.*, vol. 36, no. 6, pp. 47–56, Jun. 2003. [Online]. Available: http://dx.doi.org/10.1109/ MC.2003.1204375

[11] J. A. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Reading, MA, USA: Addison-Wesley. 2000.

[12] S. Faraj and L. Sproull, "Coordinating expertise in software development teams," *Manage. Sci.*, vol. 46, no. 12, pp. 1554–1568, 2000.

[13] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," VTT Pub. 478, Espoo, Finland, Tech. Rep., 2002, p. 107.

[14] P. Meso and R. Jain, "Agile software development: Adaptive systems principles and best practices," *Inf. Syst. Manage.*, vol. 23, no. 3, pp. 19–30, 2006.

[15] J. Erickson, K. Lyytinen, and K. Siau, "Agile modeling, agile software development, and extreme programming: The state of research," *J. Database Manage.*, vol. 16, no. 4, pp. 88–100, 2005.

[16] S. Kuppuswami, K. Vivekanandan, P. Ramaswamy, and P. Rodrigues, "The effects of individual XP practices on software development effort," *ACM SIGSOFT Softw. Eng. Notes*, vol. 28, no. 6, p. 6, 2003.

[17] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, no. 1, pp. 64–69, Jan. 2002.

[18] H. D. Benington, "Production of large computer programs," in *Proc. Navy Symp. Adv. Programm. Methods Digit. Comput. Office Naval Res.*, Washington, DC, USA, 1956, pp. 15–28.

[19] R. T. Futrell, D. F. Shafer, and L. Shafer, *Selecting Software Development Life Cycles, Quality Software Project Management*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002, pp. 101-161.

[20] B. Blanchard and W. Fabrycky, *Systems Engineering and Analysis*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2011.

[21] V. R. Basil and A. J. Turner, "Iterative enhancement: A practical technique for software development," *IEEE Trans. Softw. Eng.*, vol. SE-1, no. 4, pp. 390–396, Dec. 1975.

[22] M. Fowler. *The New Methodology*. Accessed: Dec. 12, 2004. [Online]. Available: http://www.martinfowler.com/articles/newMethodology.html

[23] M. James. (2009). *Scrum Methodology*. Accessed: Apr. 8, 2015. [Online]. Available: http://scrummethodology.com?

[24] K. Beck, "Embracing change with extreme programming," *IEEE Comput.*, vol. 32, no. 10, pp. 70–77, Oct. 1999.

[25] M. Alshayeb and W. Li, "An empirical study of system design instability metric and design evolution in an agile software process," *J. Syst. Softw.*, vol. 74, no. 3, pp. 269–274, 2005.

[26] K. Schwaber and J. Sutherland. (2011). *The Scrum Guide, the Definitive Guide to Scrum: The Rules of the Game*. Accessed: Jun. 14, 2012. [Online]. Available: http://www.scrum.org/Portals/0/Documents/ScrumGuides/Scrum_Guide.pdf?

[27] L. Rising and N. S. Janoff, "The Scrum software development process for small teams," *IEEE Softw.*, vol. 17, no. 4, pp. 26–32, Aug. 2000.

[28] J. Highsmith, *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Reading, MA, USA: Addison-Wesley, 2013.

[29] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," in *Proc. 21st Int. Conf. Inf. Syst.*, 2000, pp. 58–69.

[30] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," *IEEE Comput.*, vol. 34, no. 9, pp. 120–127, Sep. 2001, accessed: Oct. 10, 2004. [Online]. Available: http://www.jimhighsmith.com/articles/IEEEArticle1Final.pdf

[31] J. Highsmith and A. Cockburn, "Agile software development, the people factor," *IEEE Comput.*, vol. 34, no. 11, pp. 131–133, Nov. 2001, accessed: Oct. 10, 2005. [Online]. Available: http://www.jimhighsmith.com/articles/IEEEArticle2Final.pdf

[32] M. Fowler. *The New Methodology*. Accessed: Dec. 12, 2004. [Online]. Available: http://www.martinfowler.com/articles/newMethodology.html

[33] J. Highsmith, *Agile Software Development Ecosystem*. Reading, MA, USA: Addison-Wesley, 2002.

[34] P. Wendorff, *An Essential Distinction of Agile Software Development Processes Based on Systems Thinking in Software Engineering Management*. Reading, MA, USA: Addison-Wesley, 2000, p. 218.

[35] C. Chatfield and T. Johnson. (2007). *Microsoft Office Project 2007 Step by Step*. Accessed: Jul. 14, 2012. [Online]. Available: http://office.microsoft.com/en-us/project-help/a-short-course-in-project-management-HA010235482.aspx#BMtime?

[36] W. R. Duncan, "A guide to the project management body of knowledge," Project Manage. Inst., Upper Darby, PA, USA, Tech. Rep., 1996.

[37] K. Schwaber, *Agile Project Management With Scrum*. Redmond, WA, USA: Microsoft Press, 2004.

[38] W. Royce, "CMM vs. CMMI: From conventional to modern software management," *Rational Edge.*, pp. 2–9, Feb. 2002.

[39] J. Grenning, "Launching extreme programming at a process-intensive company," *IEEE Softw.*, vol. 18, no. 6, pp. 27–33, Nov. 2001.

MUHAMMAD AZEEM AKBAR received the M.Sc. and M.S. degrees in computer science from the University of Agriculture Faisalabad, Faisalabad, Pakistan. He is currently pursuing the Ph.D. degree with the Department of Software Engineering, Chongqing University, China. He has an Outstanding Academic Carrier. His research interests are global software development life cycle, requirements engineering, empirical studies, global software requirements change management, software defect prediction, and software risk management.

JUN SANG received the B.Sc. degree from Shanghai Jiao Tong University, China, in 1990, the M.E. degree and the Ph.D. degree in computer science from Chongqing University, China, in 1993 and 2005, respectively. He is currently a Professor with the School of Software Engineering, Chongqing University. His research interests include software engineering, digital image watermarking, information security, and image processing.

ARIF ALI KHAN (SM'17) received the B.S. degree in software engineering from the University of Science and Technology, Bannu, Pakistan, in 2010, the M.Sc. degree in information technology from Universiti Teknologi PETRONAS, Malaysia, and the Ph.D. degree in software engineering from the Department of Computer Science, City University of Hong Kong. He is currently an Active Researcher in the field of empirical software engineering. He has participated in and managed several software engineering related research projects. He is interested in software process improvement, 3C's (communication, coordination, and control), global software development and evidence-based software engineering. He is a Student Member of ACM.

FAZAL-E-AMIN received the B.S. degree in computer science from Hamdard University in 2003, the master's degree in information technology from Quid-i-Azam University in 2005, the master's degree in software engineering from International Islamic University in 2008, and the Ph.D. degree from the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, in 2012. He is currently serving as an Assistant Professor with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University. His research interest includes open source software, software usability, software quality, and global software development.

NASRULLAH received the M.S. degree in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2009. He is currently pursuing the Ph.D. degree in software engineering from the Chongqing University of China. His research interests are in image compression, image encryption, data hiding, machine learning, and deep learning.

SHAHID HUSSAIN received the M.Sc. degree in computer science from Gomal University, Dera Ismail Khan, and the M.S. degree in software engineering from the City University of Science and Information Technology, Peshawar, Pakistan, and the Ph.D. degree in software engineering from the Department of Computer Science, City University of Hong Kong. His research interests include the software design patterns and metrics, text mining, empirical studies, data science, and software defect prediction. He was a recipient of the Outstanding Academic Performance Award from City University of Hong Kong.

MOHAMMAD KHALID SOHAIL received the Ph.D. degree from Mohammad Ali Jinnah University. He is currently an Assistant Professor with the COMSATS Institute of Information Technology, Islamabad. He had a great teaching, research, and publication experience. He has published several high impact research journal papers. His current research interest is to target the financial issues and strategies in the industries and the role of software engineering practices.

HONG XIANG received the B.Sc. degree in mathematics from Sichuan Normal University, China, in 1984, the M.Sc. degree in mathematics and computer science from Lakehead University, Canada, in 1992, and the Ph.D. degree in mathematics from the University of Alberta, Canada, in 1998. He is currently a Professor with the School of Software Engineering, Chongqing University. His research interests include software engineering, information security, and cryptography.

BIN CAI received the B.Sc. degree from Southwest Normal University, China, in 2002, and the M.Sc. degree and the Ph.D. degree in mechanical engineering from Chongqing University, China, in 2005 and 2012, respectively. He is currently an Associate Professor with the School of Software Engineering, Chongqing University. His research interests include software engineering, optimization method, and cryptography.