IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 8, 2021, accepted December 12, 2021, date of publication December 20, 2021, date of current version January 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3136861

Agile Approaches for Cybersecurity Systems, IoT and Intelligent Transportation

YAHYA M. TASHTOUSH[1], DIRAR A. DARWEESH[1], GHAITH HUSARI[2], OMAR A. DARWISH[3], YOUSEF DARWISH[1], LUAI BANI ISSA[1], AND HUTHAIFA I. ASHQAR[4]

[1]Department of Computer Science, Jordan University of Science and Technology, Irbid 3030, Jordan
[2]Department of Computer Science, East Tennessee State University, TN 37614, USA
[3]Information Security and Applied Computing, Eastern Michigan University, Ypsilanti, MI 48197, USA
[4]Precision Systems Inc., Washington, DC 20003, USA

Corresponding author: Dirar A. Darweesh (derardrweesh@gmail.com)

ABSTRACT To adapt to the rapidly increasing vulnerabilities in software products and cyber threats that exploit them, security professionals are actively working with software developers to produce more secure systems. In software development, agile methods are increasingly adopted in critical software projects where security risks are prominent challenges. This adoption stems from the fact that agile methods are highly iterative and support delivering services and products in smaller batches which allows security professionals to seamlessly integrate software development security activities with agile methodologies. In addition, the iterative nature of agile software development encourages frequent inspections, tests, and patching of software systems to mitigate cybersecurity risks and vulnerabilities. Considering the massive growth of the Internet of Things (IoT) and Intelligent Transportation Systems (ITS) products, the challenge of software development while addressing the security and safety concerns of these devices will continue to increase. This paper presents a comprehensive and detailed review of agile software development in the context of IoT, ITS, and their cybersecurity and risk challenges. Furthermore, we provide a systematic comparison of the reviewed literature based on a set of defined criteria. Finally, we provide a broader outlook and an outline for designing future security-enhanced agile software development solutions for IoT and ITS systems.

INDEX TERMS Agile, cybersecurity, intelligent transportation, IoT, smart vehicle, software engineering.

I. INTRODUCTION

Agile approaches support the agile philosophy, which focuses on built software, customer's needs, and motivating project teams [78], [77]. Every agile approach consists of different combinations of practices, which describes how the developer does the daily work. By choosing an appropriate set of terms and practices, so every approach differs from the other. As mentioned previously, there are many agile approaches such as Crystal, Extreme Programming (XP), Scrum, Dynamic Software Development Method (DSDM), Feature Driven Development (FDD), and Kanban. We chose these six approaches because they were the most commonly used methods. Each method has its own practices, phases, roles, advantages and disadvantages. We made a comparison between these approaches. We depended on 12 criteria to examine the differences and similarities among these approaches. These criteria are Development Approach and Style, Roles, Focus, Requirements, Time, Key Features (Practices), Team Size, Communication Style, Suitable Project Size, Feedback, Software Quality, Pros and Cons. On the other hand, this paper aims to summarize the used agile approaches in the context of Cybersecurity, IoT and Intelligent Transportation Systems. A comparison between these approaches has been done either it is used or not for each of these systems.

The rest of the paper is structured as follows: literature review is discussed in section 2; Agile models are explained in section 3, which introduces information about the six used agile approaches in our research project; Discussion is presented in Section 4 that is divided into two parts. Part1 shows a comparison between these approaches according to 12 criteria, while part2 shows the studied fields in our research project which are Cybersecurity, IoT and Intelligent Transportation Systems, and summarize the used agile approaches in the context of these fields and finally Section 5 concludes this paper.

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

II. LITERATURE REVIEW

Many research papers were interested in explaining the agile approaches and the differences between them [81], [83], [91], [92], [95], [98], but none of them were interested in summarizing used techniques in the context of Cybersecurity, IoT and Intelligent Transportation Systems like what we did in our research project. Some of these papers are mentioned below.

Ajith Jerom B etal [71] provided in their paper a comparative analysis about agile application that includes distinct types of agile methods. Some of these methods are Kaban, Scrum, Lean, Extreme Programming, Adaptive Software Development (ASD), Feature Driven Development (FDD) and Dynamic Systems Development Method (DSDM), etc. Authors explained also the variation among them, and they gave recommendations for using these models. That involves where and when to use each of these models. On the other hand, the authors mentioned in their paper that there were about 14 various Kanban boards were obtainable. Their paper demonstrated how the features are presented in the boards. In order to examine their available sources on the web, a comparison with 22 application tools for applying them to virtual Kanban boards were done.

Shaikh and Abro [73] demonstrated In their paper the distinct specifications, characteristics, crucial practices, advantages and disadvantages of various approaches associated to the application product. In their paper, they utilized six approaches. These approaches involve waterfall, spiral, unified process, scrum, extreme programming and feature-driven development. On the other hand, they also briefed the determinants and cost management operators for Software Development Methodologies (SDM) during the development process of software products.

ALMandhari *et al.* [74] introduced a preview and comparative research on system development methods. Their research focused on the determinants, which may be able to improve. They made a comparison between distinct approaches, in order to explain the lineaments and limitations for each approach. The authors mentioned also in their paper that they will join between models challenges and the reasons of project failures, in order to get a hypothesis, which can be appropriate and flexible to various projects.

Hneif and Ow [75] mentioned in their paper that modern software development methods were presented during the past forty years, in order to proper the modern cultures of the software development institutes. On the other hand, they mentioned also that most software institutes these days are interested in achieving worthy product in short period of time with least costs, and within unsteady, variable environments. Agile approaches were thus presented to satisfy the new requirements of the application development institutes. Authors introduced a review of three agile methods, which involves Extreme Programming, SCRUM and Agile Modeling. Their paper also explained the variance between them, and it gave recommendations when to use them.

Dybå and Dingsøyr [83] performed a systematic revision of experimental studies of agile software development. Their research was up to and involving 2005, and its methodology specified 1,996 studies. 36 of these studies were recognized as experiential. The studies were classified into four themes. These themes include introduction and adoption, perceptions on agile methods, human and social factors and comparative studies. Their survey discussed what is presently known about the advantages and determinants of, and the force of proof for, agile approaches. The major effects for their research are a necessity for more and superior empirical studies of agile software development within a popular project agenda.

Strode [84] performed in her paper an overview of the agile approaches, involving the key publication of each approach, the main impacts on the agile techniques, and demonstrated proportional studies where comparison and analysis of methods has been carried out. Then a relative study was described, which was applied on five agile techniques for addressing the question 'what is an agile method'? A proportional analytical framework appropriate for this objective was described along with the outcomes of implementing the framework to five agile approaches, which are Scrum, ASD, XP and Crystal techniques.

Hiwarkar *et al.* [85] mentioned in their paper that traditional software development approaches are incapable to treat changing requirements during the software development process. In order to overcome this challenge, a group of software development techniques referred as ''Agile Software development methodologies'' are utilized. In their paper, they provided a comparison between these distinct agile software development approaches. They mentioned that this comparison will assist in choosing the suitable development method given a specific scenario.

III. AGILE MODELS

This section introduces information about the six agile approaches, which are used in our research project. These models involve Crystal, Extreme Programming (XP), Scrum, Dynamic System Development Model (DSDM), Adaptive Software Development, and Kanban. These models are explained below.

A. CRYSTAL

Crystal is a family of methods that can be used for different projects size, complexity, criticality, and team members [1]. Alistair Cockburn has developed the Crystal, where the method produces systems incrementally, and the time duration for each iteration should not exceed four months. It was established in 1990. Alistair Cockburn and Jim Highsmith created the Crystal family of agile methods [2], [101].

In figure 1, we note that Crystal is consists of some methods such as Crystal Clear, which is appropriate for small projects with a team size of up to 8. This method is followed by crystal yellow used for medium team size range from 10 to 20. Crystal Orange method is dedicated to large team size that ranges from 20 to 50. The last method is crystal

	Flavers	Clear	Yellow	Orange	Red
Criticality of the project	Life (L)	L6	L20	L40	L80
	Essential Money (E)	L6	E20	E40	E80
	Discretionary Money (D)	D6	D20	D40	D80
	Comfort (C)	C6	C20	C40	C80
	Team Size	Up to 8	10 to 20	20 to 50	50 to 100

FIGURE 1. Crystal's Coverage of Distinct Project Types [8].

red which is specified for substantial projects with team sizes from 50 to 100. A set of principles in the Crystal is face to face communication as stated in Cockburn's philosophy. They are focusing on flexibility based on problem characteristics, simplicity. On the other hand, they suggest using a reflection workshop to review the team's work habits. Crystal methods focus on people and communication among people rather than a process to frequently deliver working software [3], [4].

1) CRYSTAL FLAVORS

Crystal methods are represented in different colors, based on the number of team members for every crystal type [5], [106], [108]. To choose which crystal method is appropriate for the project, we must take in advance four elements: Comfort (C), Essential Money (E), Discretionary Money (D) and Life (L) [6], [107], [108].

2) POLICY STANDARDS

Several crystal policies standards and practices are applied during the development process. These policies involve the user, making user reviews after each release, using test covering strategies, making maintenance for product and method, and incrementally delivering the product [8], [109].

3) PROPERTIES OF CRYSTAL

Crystal has seven properties. These properties are: focusing on work, personal safety, osmotic communication, easy access by an expert user, automating tests within a technical environment, delivering the product rapidly, and making changes and improvements. We can apply these properties to all project sizes except the osmotic communication property, which lay within a small group of people [6], [79].

4) STRATEGIES AND TECHNIQUES OF CRYSTAL

A few strategies and techniques are suggested: early victory, walking skeleton, exploratory 360, and information radiators. Methods include methodology shaping, reflection workshop, blitz planning, Delphi estimation, daily standup, essential interaction design, Process miniature, side by side programming burn chart, and incremental re-architecture [6], [82].

5) TEAM ROLES

A large number of roles here, in which the roles and the number of teams working depends on the crystal method used. In Crystal clear, there is only one team working on the project, which includes the following roles: unit tester, documenter, sponsor, senior programmer, expert, and designer [5]. On the other hand, more than one team is working on a project according to the size in crystal orange. This is because of the complexity and big size of the project, and it includes a lot of roles in comparison with the roles used in crystal clears. These roles are database designer, technical facilitator, user interface designer, usage analyst, tester, writer, and business analyst [9].

6) CRYSTAL CYCLIC PROCESSES

Six cycles in crystal project cycle contain project ending, charting and delivery cycles. Delivery cycles include four parts iterations which are: completions, and deliveries, iteration cycle where the iteration span from 1 week to 2 months, and integration cycle that depend on the practices it runs from hours to days. The fifth cycle is the day and week cycle. The last cycle is the development episode cycle. The Crystal supports high user involvement, early, adaptability and frequent delivery of software [4], like other agile approaches.

7) ADVANTAGES OF CRYSTAL METHODS

In the Crystal, we can use a specific method for any project size. It provides technical practices and risks control and guidance about communication [7], [9].

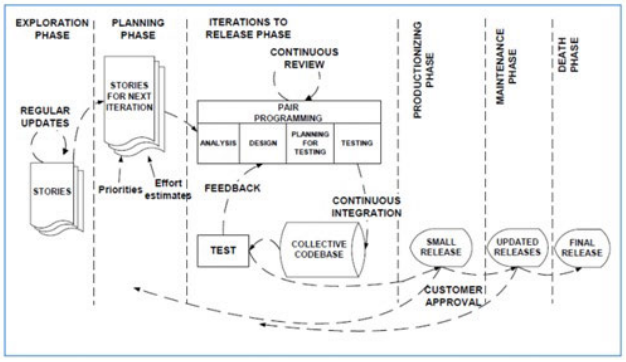


FIGURE 2. Life Cycle of Extreme Programming [14].

8) DISADVANTAGES OF CRYSTAL METHODS

It does not provide business enterprise guidance; there are no verification activities for code and design. It does not contain system validation, and it just defines two methods from 4 [7], [9].

B. EXTREME PROGRAMMING (XP)

Extreme Programming (XP) was developed by Kent Beck in 1996. It is a more flexible, low-risk, disciplined, and lightweight method as it can manage rapidly changing or vague requirements [10]. This method is more appropriate for small and medium team sizes [11], [94]. XP is a collection of principles, practices, and values that can be applied correctly [12]. The reason for calling it extreme programming is that it took practices to the extreme, which help in developing software with high quality [13]. This approach emphasizes customer satisfaction—frequent releases and feedback help in managing the defects [14].

1) XP PHASES

Figure 2 shows that XP involves 6 phases: exploration phase, planning phase, iteration to release phase, productizing phase, maintenance phase and death phase.

Exploration Phase: This phase deals with the requirement and architecture modeling of the system. In this phase, a meeting among users, developers and customers is conducted. In this phase, we define architecture, tools and requirements as stories written by customers. At the end of this phase, the developer should trust time and cost estimation for implementation. Therefore, the material should be available from user stories that can provide a good start for the first product, and the period of this phase span from few weeks to few months [14].

Planning Phase: During this phase, tasks are drawn from user stories and written on task cards. The decision about code, team size, ownership, schedule, and working hours is taken. This phase is performed in 2 parts which are iteration and release planning [15].

Release Planning: The goal of this phase is to find out the needed features and delivery schedule. Customers and developers participate in release planning meetings [13]. Then, the customer writes story cards to identify requirements. These requirements are then sorted according to their importance. Finally, a smaller set of cards is selected for the recent release. This method is considered as an iterative process [14].

Iteration Planning: In this phase, developers prepare a plan of their activities. During this phase, the programmer selects tasks to be implemented and estimates the required cost, time and effort for selected tasks [14].

Iteration to Release Phase: This phase inserts the basic activities like designing, testing [16]. This is an iterative phase, in which each iteration can span over 1 to 4 weeks. Pair programming and code refactoring are the main practices here [13].

Productionizing Phase: since XP is an incremental and iterative approach, XP delivers product in small releases. A release cycle consists of many iterations that can span from 1 to 4 weeks [17]. Therefore, the deployment of the software in this phase is in small releases. To check if the software is ready for production, the testing is performed. During this phase, the system development rate by programmers is slowed down. To know if the change goes to the next release depends on the importance of the risk [18].

Maintenance Phase: New functionality is built in this phase while keeps the old one running [13]. In the XP team introduce architectures, they have to take extra care because the system is in production, and they stop the changes that cause problems [19].

Death Phase: When the software arrives in this phase, there are two possible situations. The first one is the final release, and the second situation is called entropic death [13].

2) XP PRACTICES

The practices in XP are pair programming, short iteration, quarterly release, sit together, stories,10 minutes constructing, integration continuously, unit test, incremental design, informative workspace, whole team, slack, and active work. Also, there are eleven refrain practices: Pay as per usage practices, root cause analysis, collective code ownership, involving the customer, deployment incrementally, to continuing the team, the team shrinking, everyday deployment, one code base, testing and coding, and negotiated domain contract [20]. These practices result in five values: simplicity, quality, feedback, bravery and good communication [21].

3) XP ROLES

in XP, there are seven roles: Programmer, Customer, Tester, Tracker, Coach, Consultant and Manger [22].

4) ADVANTAGES OF XP

It supports incremental development through the system. On the other hand, it improves the quality and productivity by refactoring code to keep simplicity [23].

5) DISADVANTAGES OF XP

XP should be used for projects that require traceability and review. Involving the customer is time-consuming

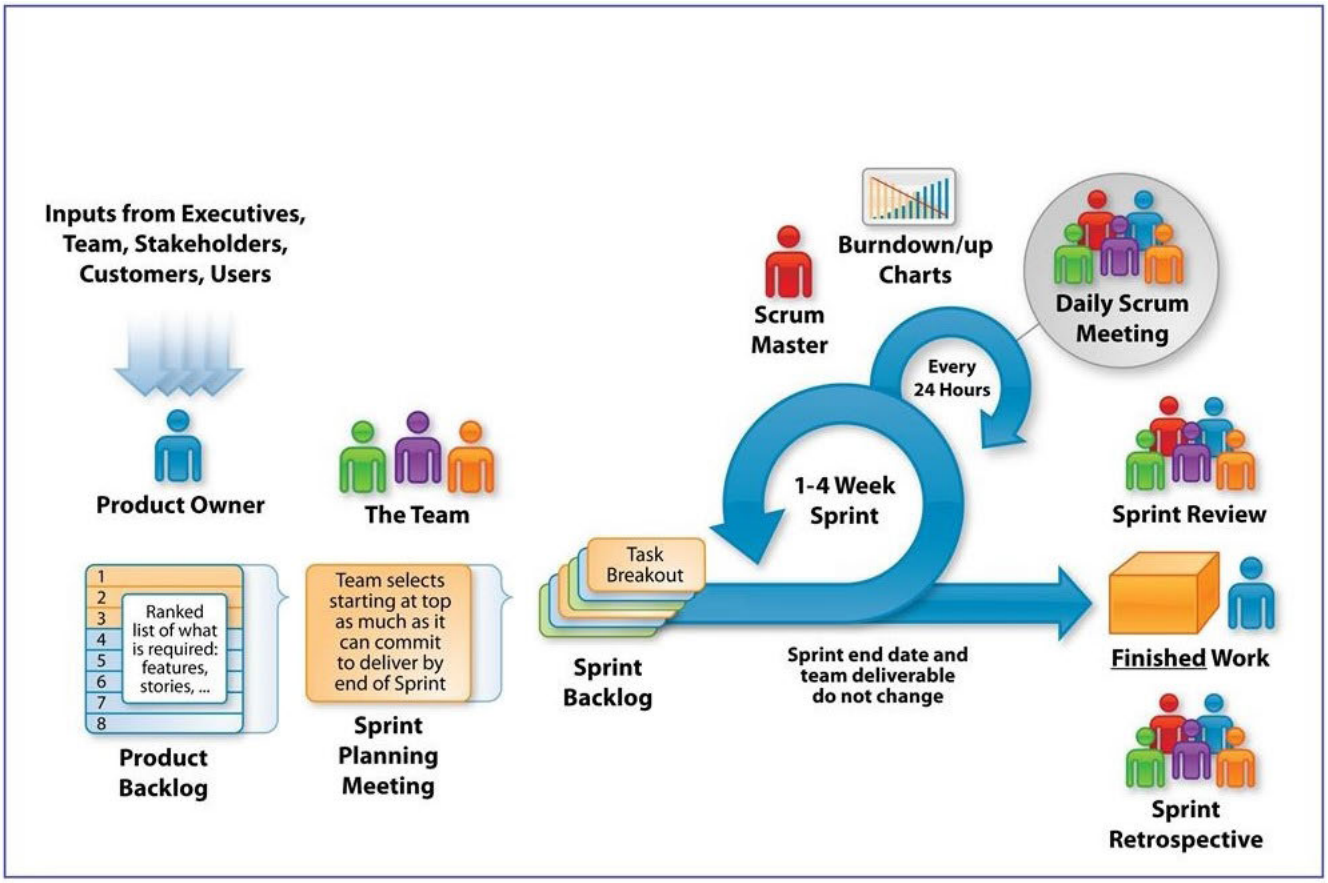


FIGURE 3. Scrum Framework [76].

and stressful. It does not support distributed teams, and some practices need extra training for members [22].

C. SCRUM

This approach is one of the most popular agile processes because it is a simple process. It also focuses on management issues to be used in any domain contrasting to other agile approaches. Beedle and Schwaber has proposed Scrum for project management. The idea of this process is to treat with faults of traditional methods. Every phase in the Scrum, we call it a sprint, and we define backlog that is a list of requirements. In this approach, the iteration is developed incrementally [14]. The scrum features are: requirements in Scrum as packets, every phase here is called sprint, which period typically 30 days, consisting of work units to satisfy the requirements, continues documentation and testing during product developing, and every iteration must be met every 24 hours. Not ready products called demos which are given to the customer with frequent deadlines [25]. Adaptation, transparency, and inspection is considered as the more powerful three points of Scrum [26]. Figure 3 below shows the events, roles and artifacts of Scrum. After that, the events are sprint planning meetings, daily standup scrum meetings, sprint reviews, and sprint retrospectives [22]. Therefore, the three roles here are the scrum master, product owner, and development team [22], [27]. The artifacts are product backlog that contains a list of total product requirements, sprint backlog, which is a collection of items chosen to the sprint and the increments, which are the summation of product backlog items done in sprint [28]–[30], [87].

1) PHASES OF SCRUM

Figure 3 explains these phases. The first phase is the outline planning, where the general objectives of the system are determined. The second phase is the development phase. It consists of a chain of the sprint cycle. Next, the excess value added to the system, which is the output for every cycle. Finally, the third phase is closing the project, where the goals and requirements are achieved and match the contract between the team and product owner, and now the wanted product is ready for release [22].

2) SCRUM WORKFLOW

Scum is an incremental and iterative approach, where the progress in a set of sprints made by projects [32]. At the end of each sprint, prioritized features are delivered. The requirements collected by the product owner from the customer are using product backlog. Here is the vision of the product specified to team. After that, the user requirements are selected, and sprint backlogs are created. The sprint includes

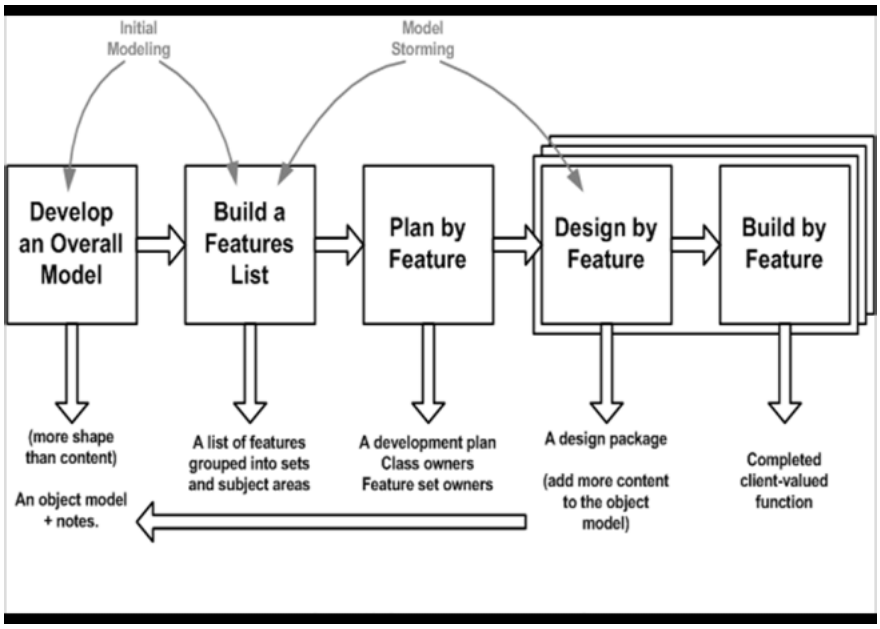


FIGURE 4. FDD Processes [19].

the goals of development as implementation, integrations and testing. Also it provides product backlog improvement and daily standup meetings. Then sprint review as a short meeting between scrum team. They focus on strategies to improve the development and clarify the positive and negative factors for the last sprint. Finally, they devise solutions to enhance vulnerabilities in consecutive sprints [31]–[34].

3) SCRUM ADVANTAGES

The first advantage is self-organization, where each employee determines the nature of his work and the team's position, and this is his decision. The second advantage is transparency when the team's vision about all things from the meeting occurs during development. Self-retrospective is the third advantage when the duties are shared among all members of the team. The fourth advantage is that the scrum process is considered simple. The fifth advantage is that it allows completing the functionalities of the team. Finally, the last advantage is that the knowledge is shredded in Scrum, and it encourages communication [21]–[23].

4) DISADVANTAGES OF SCRUM

Problems may happen in team member duties, and it never minds about engineering issues and practices [20]. Also some of the disadvantages are module Integration problems, low quality of code, disturbance in team work and shortage of Scrum training [24].

D. FEATURE DRIVEN DEVELOPMENT (FDD)

It is an approach that results in functional software. Jeff De Luca used this method in a large project when he knows that other agile approaches are not suitable for this type. The idea of this method is to handle the software development based on the requirement feature list. FDD focuses on the product with high quality through the development phases. Since this method admits late changing requirements, it is considered a highly adaptive approach [35].

1) FDD LIFE CYCLE

Figure 4 shows the five incremental, iterative successive processes. The first step is developing an overall model, where the project context and scope is determined. Then create, review and select models by experts members. The second step is building the feature list. This step is done by using documentation and models. The user may need these features, and then reviewed and confirmed, and the third step is a plan by feature, where a perfect plan is created and arranged depending on some criteria. In this process, some members of the team are involved. Finally, the fourth step, designed by feature and the fifth step, is merged into one stage [22].

2) FDD ROLES

There are six fundamental roles which are: project manager, development manager, chief programmers, domain experts, chief architect and class owner [36].

3) ADVANTAGES OF FDD

Firstly, it is considered as the more adaptive approach among others. Secondly, it focuses on quality during the overall phases. Thirdly, it can gain quick product feedback through one to four weeks [7], [9].

4) DISADVANTAGES OF FDD

The first step requires outstanding experts for modeling and designing. The second step needs assistance from other approaches because there is no requirement for supply and

risk management guidance. Finally, the final stage does not handle project relevance issues [7], [8].

E. DYNAMIC SYSTEM DEVELOPMENT MODEL

DSDM is considered as an iterative and incremental method, where the quality is a very significant value. The idea of this method was in 1994 by practitioners in the UK. Later it became an approach for rapid applications. The central concept of DSDM is to determine the time and resources of the project and later detect the functionality [35]. It supports the formula 80 percent of the solution in 20 percent of the time. This approach is more appropriate for the system, where requirements happened in a short span. It aims to deploy the product on time and within budget while adapting to new changes or conditions [37].

1) PRINCIPLE AND PRACTICES OF DSDM

Good test covers the system from start to finish the project. DSDM is likely to choose frequent and short delivery. It involves the user during project phases. The software is developed concerning business needs. It authorizes the team for decision making, delivering the product incrementally and develops the solution iteratively to be developed correctly and on time. The needs of the system at high level, while the developing changes in the project are reversible. It uses good collaboration between stockholders and team members. It focuses on quality constantly. Plans enforce control, and to gain feedback, must continuously communicate [22], [23], [38], [39].

2) DSDM LIFE CYCLE

It is shown in figure 5 that five phases are done in DSDM, which are: 1st phase is feasibility study phase. In this phase, we want to check if the DSDM is suitable for development, so we study the project. A plan and report are the results of the first phase. The second phase donates the business study stage: The most prominent features of this step are the interviews among the development team and business experts to produce a list of the required functions and determine the user's requirements. The main results here are the prototyping plan, system architecture and ER diagram. The third phase is functional model iteration. This step is incremental and iterative for coding, prototyping and analyzing. Determining the developed software is the fundamental goal for this phase. The fourth phase is the design and build phase. In this phase, the requirements defined in the third phase are coded, published, and tested iteratively. The main results here are requirements and tested software. The fifth phase is the implementation phase. Here the software is produced and then handled with developed manual to the user. Later we checked if the resulting software is satisfying the user needs or not. If not, we start from begging. The exchanging of information takes place using documents. In this phase, product developing and designing occurs incrementally. Finally, it is documented to be delivered after satisfying the needed requirements [22], [35], [93].

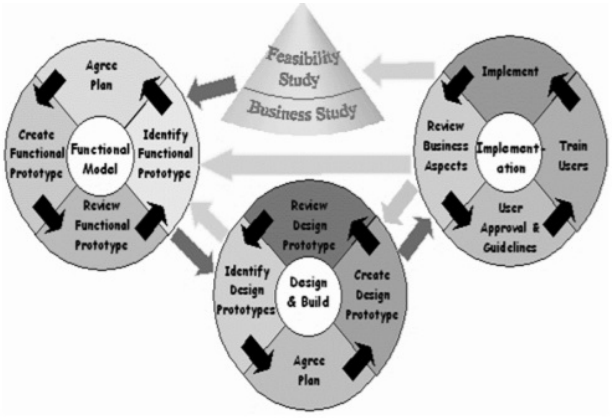


FIGURE 5. DSDM Phases [19].

3) DSDM ROLES

Main roles are: advisor user, visionary, developers, technical coordinator, executive sponsor, and ambassador user [22], [35].

4) ADVANTAGES OF DSDM

It offers rapid application development (RAD) with the integral and principle of agile. Also, it mixes the best features and practices with other methods [7], [8].

5) DISADVANTAGES OF DSDM

firstly many roles in DSDM so that some administration problems may occur. Secondly, it does not consider the importance for the project. Thirdly, it does not offer guidance about the length of iteration and team size [7], [8].

F. KANBAN

In 2004, David J.Anderson created this approach. Kanban consists of two syllables ''kan'' means visual ''Ban'' means cards, so Kanban focuses on a visualized workflow, where work is divided into smaller tasks and displayed on the board. This board consists of columns, visual cards (the main feature of any Kanban board cards, stickies), and a WIP limit. It assists in maintaining transparency [40]. The main goal of this approach is to protect the team from the never-ending tasks assigned by management. It tries to achieve continuous adaption and development like other approaches, where there is little or no fight to change [41]. Here, it suggested refraining the team from developing software features before they are ordered because it is based on just in time hypothesis, especially when treating software development [42].

1) KANBAN PRINCIPLES

The quality must be included within the development, maximizing tasks that are executed at any time in the process, attracting value through the developing, it focuses on the visibility of developing, it uses static backlog, and concentrates on increasing the productivity [43].

2) KANBAN PRACTICES

the first is to visualize the workflow, where the work must be visible for all team members, to provide a positive effect on the quality of the product [90]. This clarifies visualization using the Kanban board, but invisibility has a negative impact on the implementation of the project [44], [45], [80]. The second is to practice a limited work in progress. This practice indicates decreasing the number of tasks or features while developing. It is worth noting that if we reduce the number of tasks of work, then the time to deploy the product is also decreased [44], [46], [47]. The third is to practice is managing and measuring flow, where terminology measuring flow, when every state in work is controlled and reported. So the movement will be smooth and convenient with a positive effect in reducing the risk, avoiding the cost of delay and creating high value in time [44]. The fourth practice is making policies explicit. Since most of the organizations have policies that take into account the different types of work, which have to deal with, so Kanban teams explicitly establish policies for their consistent application [45]–[47]. And the fifth practice is implementing feedback to compare predictable outcomes and real outcomes using reviews and meetings. Then making needed adjustments [44], [46]. Finally, the sixth practice is continuous improvement. In this practice, the clear and shared understanding of business theories, process, and risk help members of the team to understand the problem, and suggest improvements, so the team must use the mind of Kaizen, which is a primary part of efficient use of this approach [48], [49].

3) ADVANTAGES OF KANBAN

Firstly, the workflow is visualized clearly, giving the team members comfort. Secondly, it promotes the team for motivation. Thirdly, the last results in Kanban improve customer satisfaction and confidence. Next is the fourth step, which implementing Kanban in developing means taking on practices for improving the current process. The fifth step that Kanban has some principles and rules that makes it easy to implement [50]. The sixth step has fewer budget and time requirements. The seven-step is early delivery of the product, which is allowed. The eighth step comes with process policies that are transcribed. Finally, step number nine, in Kanban, the improvement of the process is continuously [54].

4) DISADVANTAGES OF KANBAN

Firstly, the unwillingness of the developer to use the measurement. Secondly, the developers may have a negative effect on their focus from the flexibility. Thirdly, the project may ruin because of the poverty of business analysis. Fourthly, the roles are not defined in kanban. Step number five, the collaboration among the Kanban team may affect the success negatively. Then step number six, where the main disadvantage of Kanban is that it was used for small user stories, and it neglects the large ones. Therefore, the result is asymmetric task distribution [52], [54].

IV. DISCUSSION

This section is consists of two parts. Part1 shows a comparison between these agile approaches according to 12 criteria. part2 explains the studied domains in our research, which are Cybersecurity, IoT and Intelligent Transportation Systems, and summarize the used agile methods in the context of these domains.

A. AGILE DEVELOPMENT MODELS COMPARISON

Since we mentioned previously, that there are many agile methods as we explained. Each approach has its drawbacks and pros, and thus makes the agile approaches are appropriate for different sizes and types of business projects. Team size, the development environment, requirements, quick feedback, development style, roles, focusing, time, key features, practices, communication style, advantages and disadvantages are some criteria of agile software development approaches that enable us to select appropriate projects.

Since any agile method has its own flaws, it cannot be used for all projects, especially complex, large and crucial projects [51], [53], [103].

To eliminate the defects of agile methods, coding practitioners create new methods by combining two or more agile methods. Thereby, it crushes the defects and improving the positives. Table 1 presents a comparison among six agile approaches discussed in section3.

Many facts exposed about the six agile approaches mentioned in Table 1. Therefore, we used twelve criteria for the comparison among them.

It is observed that the crystal, XP, Scrum, FDD, and DSDM models are iterative and incremental development approaches except for Kanban, which is a straightforward iterative approach. We note that these approaches are adaptive. Concerning roles, each method has roles attached to it. As noted, there are large numbers of roles in crystal-like project sponsor, senior designer, software documenters and unit testers, etc., but there are no defined roles in Kanban. Each method focuses on the things that may differ, and those things that it focuses on may be similar. For example, Crystal focuses on people and communication among them. XP emphasis engineering issues. Scrum emphasis management issues. If we combine Scrum and XP, the output will have a positive effect on productivity [97]. FDD and DSDM focus on quality during the development. Kanban focuses on workflow visualization. In which the requirements in each method differ from the other. In XP, they are in the form of stories in FDD. They are in the form of features, but in Scrum, the development work is partitioned as ''packets''.

Regarding the time, XP, Scrum, and FDD each have a specific period of time. The iteration in Scrum is from two to four weeks, whereas XP iterations are from one to two weeks. DSDM takes the Pareto principle into account with respect to time, unlike Scrum, where each iteration from two to four weeks. In FDD, unlike Scrum, the iteration takes two weeks of time. With respect to time in Kanban, it is optional.

AQ:6 **TABLE 1. Comparison among agile software development models.**

Practices	Crystal	XP	Scrum	FDD	DSDM	Kanban
Development Approach	Incremental/ Iterative [2].	Increments are iterative [17].	Iterative increments [14] [31].	Short incremental iterations [35].	Incremental iterative [35].	Short iterations [44].
Development Style	Adaptive [4]	Adaptive [14].	Adaptive [26].	High adaptive [35].	Adoptable framework [8].	
Roles	There are numerous roles in Crystal [5].	Seven roles programmer, customer, tester, tracker, coach, consultant and manager [22].	Scrum master, product owner and scrum team [27].	Project manager, chief architect, development manager, domain expert, class owner and chief programmer [36].	Developers, technical coordinator, ambassador, user advisor, visionary and executive sponsor [22] [35].	Not define any role [53].
Focus	On people and communication among them [2].	Emphasis on engineering issues [55].	Emphasis on management issues [20].	On quality during the developing [35]. Mainly on design and building [35].	• On business needs [23]. • On quality [23].	On a visualized workflow [40].
Requirements	Allow "maneuverability" based on problem characteristics. [25].	As user stories [14].	As "packets" [25].	Requirements as "features" [22].	Requirements are baseline at a high level [22] [38].	Visual Cards – the main feature of any Kanban board is stickies, cards, etc [40].
Time	Frequent delivery [4].	1-6 weeks [13] [14] [17].	2- 4 weeks [22].	Two days to two weeks [22].	80 percent solution in 20 percent time [37].	Continuous delivery [53].
Key features (Practices)	It can be applied for different projects size, so it is considered as a family of methods that uses the colors to represent it [3] [7].	User stories [14], refactoring [13] and pair programming [13].	Sprint [29], product backlog [28], sprint backlog [29] and scrum meetings [22].	• It has the ability to be applied on larger team size [4]. • In FDD the practices is very specific and contain 5 processes [22]. • Developing is architectural. • Developing the overall model. • Building the feature list • Planning by feature. • Designing by feature. • Building by feature [22].	• It is an approach for the rapid applications [35]. • It focuses on quality constantly [23]. • The work in DSDM can be shared with more than one team [22] [35]. Prototyping, feasibility and business study [22] [35].	Kanban has six practices which are: visualizing the workflow, limiting work in progress and continuous improvement [44] [43][45][46].

TABLE 1. ***(Continued.)*** **Comparison among agile software development models.**

Team Size	From 8 to larger [5] [6].	Small team, 2-10 [75].	5-9 (Multiple teams of less than 10 members) [56] [57].	From 4 to 20, more than one team [4].	From 2 to 10 [57].	Small to medium [57].
Communication Style	Emphasize on communication as face to face [4].	Oral, through standup meetings [13] [14].	Oral, through Scrum meeting [25].	Meeting in business phase [22] [35].	The exchange of information takes place using documents [35].	Standup meeting [44].
Suitable Project size	For small and medium sizes [57].	For small size [57].	For all size [57].	For large size [57].	For all size [57].	For all size [57].
Feedback	It suggests the use of reflection workshop to review the process [4] [51].	Uses feedbackthat span on different time scale from second to months [14] [22] [23].	Provides feedback on sprint increment [34].	In FDD a quick feedback span from 1 to 4 weeks [9].	• Communicate continuously to get feedback [23]. • The user's feedback is used to enhance the system iteratively [22] [35].	Kanban needs feedback loops to work [44].
Software Quality	Doesn't not pay much attention to the qualityof the product compared to other methods	XP is focusing on improving quality [55].	The sprint evaluation gathering is the most important exercise for quality improvement [55].	Its teams specially focus on quality throughout the development phases [9].	Focusing on improving quality [23].	When we want to compare it with Scrum, it fixates mostly on performance improvement in Kanban [55].
Pros	In Crystal, we can use specific method for any size of projects. It provides a technical practices, risk control andguidance about communication [7] [9].	It supports incremental development through the system. It improves the quality and productivity by refac-toring of code to keep simplicity [23].	• Priorities are specified by product owner. • Decision making is taken by the team. • It does not care of documentation. • It supports frequent updating [53].	• It is considered as the more adoptive approach among other. • It focuses on quality during the overall phases. • It can gain quick product feedback just through 1 to 4 weeks [7] [9].	It offers rapid application development (RAD) with integral and with principle of agile. Also it mixes the best features and practices with other methods [7] [8].	1st the workflow is visualized clearly, so it gives comfortable for team member. 2nd it promotes the team for motivation. 3rd last results in Kanban raise customer satisfaction and confidence. 4th implementing Kanban in developing means taking on practices for improving the current process. 5th Kanban has some principles and rules that makes it easy to implement [50]. 6th it has less budget and time requirements. 7th early delivery of the product is allowed. 8th process policies are transcribed. 9th in Kanban, the improvement of process is continuously [53].

TABLE 1. *(Continued.)* **Comparison among agile software development models.**

Cons	It doesn't provide business enterprise guidance; there are no verification activities for code and design.It doesn't contain system validation, and it just define two methods from four [7] [9].	XP should be used for projects that require traceability and review. Involving the customer is consuming the time, and it is stressful. It doesn't support distributed teams, and some practices need extra training for members [22].	• It is difficult to control the cost of changes. • It may not be suitable for large teams. • It requires expert team members [53].	• It requires outstanding experts for modeling and designing. • It needs assistance from other approaches because there is no requirements supply and risk management guidance. • It does not handle issues regarding to project relevance [7] [8].	1st many roles in DSDM, So some administration problems may occur. 2nd it doesn't consider importance for project. 3rd it doesn't offer guidance about length of iteration and team size [7] [8].	1st unwillingness of the developer to use the measurement. 2nd the developers may have negative effect on their focus from the flexibility. 3rd the project may ruins because of poverty of business analysis. 4th the roles are not defined in kanban. 5th the collaboration among Kanban team may affect negatively on the success. 6th the main disadvantage of Kanban that it was used for small user stories, and it neglects the large ones. Therefore, the result is asymmetric task distribution [51] [53].

Each approach has different features and practices. The primary features of Crystal are: It can be applied for different project sizes, so it considers a family of methods that uses the colors to represent it. In XP, there are many practices such as user stories, refactoring, pair programming, etc. The primary scrum practices are a sprint, product backlog, sprint backlog and scrum meetings. The fundamental features of FDD are: Firstly, it can be applied to larger team size. In FDD, the practices are particular and contain five processes. Also FDD development is architectural. DSDM features are an approach for rapid applications. It focuses on quality constantly, and the work in DSDM can be shared with more than one team. The primary practices of it are: prototyping, feasibility and business study. But Kanban has six practices: visualizing the workflow, limiting work in progress, and continuous improvement. Concerning the team size, in Crystal, it is from eight to larger.

On the other hand, in XP the team is small, from two to ten. In Scrum, the size is from five to nine. In FDD, the range is from four to twenty members. In DSDM, the team is independent, and the size is ranging from two to ten members. In Kanban, the team size is range from small to medium. The communication style within the process for all approaches here is meeting. For example, In XP, Scrum and Kanban the communication is meetings without a chair. Crystal emphasizes communication like face to face, but in DSMD, the exchange of information takes place using documents. For suitability of project size criteria, Crystal is better for small and medium project size. Scrum, Kanban and DSDM are suitable for all project types, and XP is more suitable for smaller projects, but FDD for larger projects. The feedback is done to every approach in this paper; in Crystal, it suggests using a reflection workshop to review the process. XP uses feedback that span on the different time scale from second to months. Scrum provides feedback on sprint increment. In FDD, a quick feedback spans from one to four weeks. In DSDM, communications are continuously getting feedback, and the Kanban needs feedback loops to work. Concerning software quality, Crystal does not pay much attention to the quality of the product compared to other methods, where product quality is the primary focus in DSDM. Kanban and XP also are focusing on improving quality. The primary focus of FDD is to provide quality outputs throughout all stages of the development process. In Scrum, the most important exercise for quality is Sprint evaluation gathering.

B. AGILE MODELS APPLICATIONS

In this part, we demonstrate the studied fields in our project, which are Cybersecurity, IoT and Intelligent Transportation

TABLE 2. Comparison among agile software development models.

Agile Methods	Cybersecurity	Internet of things (IoT)	Intelligent Transportation (ITS)
Crystal	N/A	N/A	N/A
XP	Yes [35]	Yes [61]	Yes [62]
Scrum	Yes [19]	Yes [61]	Yes [62]
FDD	Yes [59]	Yes [64]	N/A
DSDM	Yes [63]	Yes [64]	N/A
Kanban	Yes [60]	Yes [65]	N/A

Systems, and we brief the used agile methods in the context of these domains. We made a comparison table for these agile approaches. Table 2 shows these approaches either it is used or not for each of these fields.

1) CYBERSECURITY

Cybersecurity is defined as methods utilized to protect the cyber environment of a user or institution. These techniques control the set of methods used for conserving the safety of networks, data and applications from prohibited access. Cybersecurity may also indicate to as information technology security. The importance of cybersecurity has been increased because of the large dependence on computer systems. These systems are involving televisions, smartphones and several small devices, which shape the Internet of Things [70], [112], [113], [104]. Security threats to computing information systems have become more advanced and powerful, and therefore a rapid response becomes a necessity to mitigate these threats. As a result, conventional methods to IT security may not work. Agile techniques offer a framework for quick reaction in a dynamic environment [96], [111], [114]. As agile practices are used to enhance quality according to their short iterations and rapid releases, they can also improve IT security due to their adoption [71], [89], [113]. This research mentions the agile approaches, which has been used in the cybersecurity field. Scrum is considered one of the most common and effective agile development techniques. Anyhow, like other agile techniques, Scrum has been criticized cause of the shortage of support for building secure software. Thus, research has been conducted by integrating a security backlog (SB) to Scrum. The results of the authors' research come up with building secure applications by utilizing an agile Scrum model [19], [116].

On the other hand, XP agile approach has been used within a framework called FISA-XP. The authors developed this framework for building secure software. The process was done by combining security activities with the main activities of XP agile approach according to the level of their agility [35], [105], [117]. An improved Feature Driven Development (FDD) model for secure system development has been built. This model has been built to overcome the limitations when using the existing FDD agile model alone [59], [110]. For any business environment, which is changing continuously, there is a necessity to control security risks. Security risk administration can be automated and associated with the processes in a software development institute by utilizing an agile technique. Research authors used a technique, which is called Kanban for security risk management [60].

2) INTERNET OF THINGS (IoT)

The Internet of things indicates a network to link anything with the Internet depending on specified protocols by using information sensing tools to perform information interchange and communications for obtaining positioning, smart recognitions, tracing, control, and management [69], [99], [102]. The Internet of Things (IoT) is promoting from a technological buzz-phrase to provide private and organizational certainty with associated devices among the Internet [82], [100]. Although the literature is concentrating on the technological ways of IoT, research concentrating on the development operation of IoT solutions is still uncommon [61]. This research is focused on mentioning agile methods, which used in the Internet of Things domain. Agile approaches are considered as a natural fit for the requirements related to the development of IoT products. This operation is extending from the flexible refinement of solutions to merge the expertise of developers. These two areas of concern were brought together by researching the adaptation of agile methods in institutions, which build and spread industrial IoT products. The research results detected good marbles into the specific adaption of the agile development process, which is called Scrum in the IoT industrial field. Agile methodologies like Scrum or Extreme Programming (XP) are considered 'lightweight', iterative methods for the development of information systems (IS)). These approaches are not just a great fit for IoT development requirements but are increasingly implemented in practice [61], [115].

On the other hand, another research in [64] has been conducted among four selected small-medium (SME) institutes. These companies were interested primarily in developing

IoT solutions. The outputs of the research referred that all these companies were using agile methodologies for developing IoT products. These methodologies are extreme programming (XP), Feature Driven Development (FDD) and Dynamic System Development Methodology (DSDM). Anyhow, the most utilized agile method framework is the Scrum technique [64]. The other agile approach used in the Internet of Things (IoT) is Kanban. This agile approach has been used for the administration and monitoring of IoT sensors [65].

3) INTELLIGENT TRANSPORTATION SYSTEMS (ITS)

Transportation is considered a leading power behind the evolution of economic and well-being levels for all people around the world. Therefore, a driving transport system is an essential element for urban development. However, inadequate urban transport systems have adverse effects. These effects include crowding, delays, accidents, elevated consumption of energy, weak resources productivity, contamination, society cutoff and inconvenient arrival to the service. For this reason, it is very necessary to develop a fascinating transport system, easy to use in terms of safety, accuracy, time of traveling, and comfort. This system can be achieved by using an intelligent transportation system (ITS) [86], [88]. On the other hand, as COVID-19 pandemic has a massive universal disturbance with large economic, ecological and social effects throughout the world. It is very obvious that people around the world are suffering from this pandemic. However, Intelligent Transportation System has a significant function in relieving the bad impacts of the current pandemic and future devastating events [66]. The Agile approach provides the flexibility to develop progressing transportation systems and can perform true end-to-end supply chain abilities [67], [68], [80]. In this research project, we are interested in presenting the agile approaches, which had been used in the Intelligent Transportation field. In [62] authors demonstrated the efficiency of using the agile approach and change management with focusing on quality based on scenario-driven regression simulation. This scenario has been implemented in the CarOLO project to build an independently driving vehicle to contest in the 2007 Defense Advanced Research Projects Agency (DARPA) Urban Challenge program. The agile development process, which they utilized, consists of essential parts from XP and Scrum.

Another approach used in the cybersecurity field is called Dynamic System Development Method (DSDM). DSDM has been criticized when using alone since it lacks to security practices in its stages. CBR-DSDM model had been built to cover the security issues for a whole development life cycle of DSDM stages [63].

As we notice in table 2, all agile approaches has been applied on Cybersecurity and Internet of things (IoT) domains except Crystal model. On the other hand, we notice that on Intelligent Transportation systems (ITS) the only used agile methods were XP and Scrum. We notice also that Crystal model has not been applied in any of these domains.

V. CONCLUSION

In this paper, we have provided a systematic review of agile software development methods in the literature, highlighting the principles, roles, stages, practices, events, and the pros and cons of each method. Then, we conducted a comparison review between these methods based on twelve carefully selected criteria (Team Size, Development Environment, Requirements, Quick Feedback, Development Style, Roles, Focusing, Time, Key Features and Practices, Communication Style, Advantages and Disadvantages), in order to compare and contrast the cybersecurity aspects of these methods to help guide selecting the most appropriate agile method for a specific project based on its characteristics.

In addition, the provided comparative review can clearly indicate which agile methodology should fit a particular development environment, such as Inter of Things (IoT) and Intelligent Transportation Systems (ITS). This survey is intended to serve as a comprehensive and valuable source for future directions to enhance the cybersecurity integration with agile software development and in selecting the most appropriate agile method for a software development project based on its development environment and characteristics.

REFERENCES

[1] J. Highsmith, *Agile Software Development Ecosystems*, vol. 13. Reading, MA, USA: Addison-Wesley, 2002.
[2] J. A. Livermore, "Factors that significantly impact the implementation of an agile software development methodology," *J. Softw.*, vol. 3, no. 4, pp. 31–36, 2008.
[3] A. Vresk, "Agilne metode za upravljanje projektima," M.S. thesis, Faculty Org. Inform., Dept. Org., Univ. Zagreb, Zagreb, Croatia, 2020.
[4] F. Anwer, S. Aftab, U. Waheed, and S. S. Muhammad, "Agile software development models TDD, FDD, DSDM, and crystal methods: A survey," *Int. J. Multidisciplinary Sci. Eng.*, vol. 8, no. 2, pp. 1–10, 2017.
[5] B. Boehm, "A survey of agile development methodologies," Laurie Williams, Tech. Rep., 2007.
[6] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams: A Human-Powered Methodology for Small Teams*. Reading, MA, USA: Addison-Wesley, 2004.
[7] E. Mnkandla and B. Dwolatzky, "Agile software methods: State-of-the-art," Agile Softw. Develop. Qual. Assurance, USA, Tech. Rep., 2007, doi: 10.4018/978-1-59904-216-9.ch001.
[8] M. Ibrahim, S. Aftab, B. Bakhtawar, M. Ahmad, A. Iqbal, N. Aziz, M. S. Javeid, and B. N. S. Ihnaini, "Exploring the agile family: A survey," *Int. J. Comput. Sci. Netw. Secur.*, vol. 20, no. 10, p. 163, 2020.
[9] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," VTT Publ., Espoo, Finland, Tech. Rep. 478, 2002, pp. 3–107.
[10] J. Newkirk, "Introduction to agile processes and extreme programming," in *Proc. 24th Int. Conf. Softw. Eng. (ICSE)*, 2002, pp. 695–696.
[11] E. Mnkandla and B. Dwolatzky, "A survey of agile methodologies," *Trans. SA Inst. Electr. Eng.*, vol. 3, pp. 236–247, Dec. 2004.
[12] E. R. Mahajan and E. P. Kaur, "Extreme programming: Newly acclaimed agile system development process," *Int. J. Inf. Technol.*, vol. 3, no. 2, pp. 699–705, 2010.
[13] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, MA, USA: Addison-Wesley, 2000.
[14] S. Alsaqqa, S. Sawalha, and H. Abdel-Nabi, "Agile software development: Methodologies and trends," *Int. J. Interact. Mobile Technol.*, vol. 14, no. 11, p. 246, Jul. 2020.
[15] T. Saeed, "Mapping formal methods to extreme programming (XP)—A futuristic approach," *Int. J. Nat. Eng. Sci.*, vol. 8, no. 3, pp. 35–42, 2014.
[16] M. C. Paulk, "Extreme programming from a CMM perspective," *IEEE Softw.*, vol. 18, no. 6, pp. 19–26, Nov. 2001.
[17] R. Juric, "Extreme programming and its development practices," in *Proc. 22nd Int. Conf. Inf. Technol. Interfaces*, Jun. 2000, pp. 97–104.

[18] T. Dudziak, "Extreme programming an overview," *Methoden Werkzeuge Softw. Produktion WS*, pp. 1–28, Jan. 2000.

[19] I. Ghani, Z. Azham, and S. R. Jeong, "Integrating software security into agile-Scrum method," *KSII Trans. Internet Inf. Syst.*, vol. 8, no. 2, pp. 646–663, 2014.

[20] L. Williams, "Agile software development methodologies and practices," *Adv. Comput.*, vol. 80, pp. 1–44, Jan. 2010.

[21] D. Cohen, M. Lindvall, and P. Costa, "Agile software development," DACS SOAR, MD, USA, Tech. Rep., 2003.

[22] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," 2017, *arXiv:1709.08439*.

[23] *What is DSDM*. Accessed: Jul. 20, 2021. [Online]. Available: https://www.agilebusiness.org/page/whatisdsdm

[24] R. Akif and H. Majeed, "Issues and challenges in scrum implementation," *Int. J. Sci. Eng. Res.*, vol. 3, no. 8, pp. 1359–1362, 2012.

[25] *From slides are designed to accompany Software Engineering: A Practitioner's Approach*. 7/e (McGraw-Hill, 2009) Slides Copyright 2009 by Roger Pressman. Accessed: Jul. 20, 2021. [Online]. Available: https://slideplayer.com/slide/4159859/

[26] J. Sutherland and K. Schwaber, "The scrum papers: Nuts, bolts, and origins of an agile process," Scrum Training Inst., USA, Tech. Rep., 2007.

[27] K. Bhavsar, V. Shah, and S. Gopalan, "Scrum: An agile process reengineering in software engineering," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 3, pp. 840–848, 2020.

[28] S. Ashraf and S. Aftab, "Latest transformations in scrum: A state of the art review," *Int. J. Modern Educ. Comput. Sci.*, vol. 9, no. 7, pp. 12–22, Jul. 2017.

[29] E. D. Nuevo, M. Piattini, and F. J. Pino, "Scrum-based methodology for distributed software development," in *Proc. IEEE 6th Int. Conf. Global Softw. Eng.*, Aug. 2011, pp. 66–74.

[30] S. Ashraf and S. Aftab, "Pragmatic evaluation of IScrum & scrum," *Int. J. Modern Educ. Comput. Sci.*, vol. 10, no. 1, pp. 24–35, Jan. 2018.

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

[32] A. M. AlMutairi and M. R. J. Qureshi, "The proposal of scaling the roles in scrum of scrums for distributed large projects," *Int. J. Inf. Technol. Comput. Sci.*, vol. 7, no. 8, pp. 68–74, Jul. 2015.

[33] A. A. Albarqi and R. Qureshi, "The proposed L-Scrumban methodology to improve the efficiency of agile software development," *Int. J. Inf. Eng. Electron. Bus.*, vol. 10, no. 3, pp. 23–35, May 2018.

[34] A. I. Khan, M. R. J. Qureshi, and U. A. Khan, "A comprehensive study of commonly practiced heavy light weight software methodologies," 2012, *arXiv:1202.2514*.

[35] Sonia, A. Singhal, and H. Banati, "FISA-XP: An agile-based integration of security activities with extreme programming," *ACM SIGSOFT Softw. Eng. Notes*, vol. 39, no. 3, pp. 1–14, Jun. 2014.

[36] S. Goyal, "Major seminar on feature driven development," *Jennifer SchillerChair Appl. Softw. Eng.*, Aug. 2008. [Online]. Available: https://docplayer.net/8768970-Major-seminar-on-feature-driven-development.html

[37] B. G. Tavares, C. E. S. da Silva, and A. D. de Souza, "Practices to improve risk management in agile projects," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 29, no. 03, pp. 381–399, Mar. 2019.

[38] M. Stoica, M. Mircea, and B. GHILIC-MICU, "Software development: Agile vs. Traditional," *Inf. Economica*, vol. 17, no. 4/2013, pp. 64–76, Dec. 2013.

[39] A. Sani, A. Firdaus, S. Ryul Jeong, and I. Ghani, "A review on software development security engineering using dynamic system method (DSDM)," *Int. J. Comput. Appl.*, vol. 69, no. 25, pp. 33–44, May 2013.

[40] D. J. Anderson and S. Roock, "An agile evolution: Why Kanban is catching on in Germany and around the world," *Cutter IT J.*, vol. 24, no. 3, p. 6, 2011.

[41] D. I. K. S. berg, A. Johnsen, and J. Solberg, "Quantifying the effect of using Kanban versus SCRUM: A case study," *IEEE Softw.*, vol. 29, pp. 47–53,2012.

[42] A. Bolaji, "A cross-disciplinary systematic literature review on Kanban," M.S. thesis, Dept. Inf. Process., Univ. Oulu, Oulu, Finland, 2015. [Online]. Available: https://jultika. oulu. fi/files/nbnfioulu-201502111073.pdf

[43] 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. 2017.

[44] D. J. Anderson, "Kanban," Tech. Rep., Blue Hole Press, Sequim, WA, USA, 2010.

[45] M. O. Ahmad, J. Markkula, and M. Oivo, "Kanban in software development: A systematic literature review," in *Proc. 39th Euromicro Conf. Softw. Eng. Adv. Appl.*, Sep. 2013, pp. 9–16.

[46] A. Shalloway, "Demystifying Kanban," *Cutter IT J.*, vol. 24, no. 3, p. 12, 2011.

[47] B. Estacio, R. Prikladnicki, M. Mora, G. Notari, P. Caroli, and A. Olchik, "Software kaizen: Using agile to form high-perfomance software development teams," in *Proc. Agile Conf.*, Jul. 2014, pp. 1–10.

[48] 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, 2015.

[49] N. Kirovska and S. Koceski, "Usage of Kanban methodology at software development teams," *J. Appl. Econ. Bus.* vol. 3, no. 3, pp. 25–34, 2015.

[50] K. N. Rao, G. K. Naidu, and P. Chakka, "A study of the Agile software development methods, applicability and implications in industry," *Int. J. Softw. Eng. its Appl.*, vol. 5, no. 2, pp. 35–45, 2011.

[51] C. Matthies, "Scrum2kanban: Integrating Kanban and scrum in a university software engineering capstone course," in *Proc. 2nd Int. Workshop Softw. Eng. Educ. Millennials*, Jun. 2018, pp. 48–55.

[52] D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," 2014, *arXiv:1409.6600*.

[53] S. P. Roger and R. M. Bruce, *Software Engineering: A Practitioner's Approach*. New York, NY, USA: McGraw-Hill, 2015, p. 52.

[54] K.Mar and K.Schwaber. (2002). *Scrum With XP*. Accessed: Jul. 20, 2021. [Online]. Available: https://www.informit.com.

[55] S. M. Saleh, S. M. Huq, and M. A. Rahman, "Comparative study within scrum, Kanban, XP focused on their practices," in *Proc. Int. Conf. Electr., Comput. Commun. Eng. (ECCE)*, Feb. 2019, pp. 1–6.

[56] F. Anwer, S. Aftab, S. S. Muhammad, and U. W. Waheed, "Comparative analysis of two popular agile process models: Extreme programming and scrum," *Int. J. Comput. Sci. Telecommun.* vol. 8, no. 2, pp. 1–7, 2017.

[57] *Agile Methodology*. Accessed: Jul. 20, 2021. [Online]. Available: https://www.toolsqa.com/agile/agile-methodology/

[58] A. Firdausa, I. Ghanib, and S. R. Jeong, "Secure feature driven development (SFDD) model for secure software development," in *Proc. Int. Conf. Innov., Manage. Technol. Res.*, Malaysia, vol. 22, 2013, pp. 546–553.

[59] V. Dorca, R. Munteanu, S. Popescu, A. Chioreanu, and C. Peleskei, "Agile approach with Kanban in information security risk management," in *Proc. IEEE Int. Conf. Automat., Quality Test., Robot. (AQTR)*, May 2016, pp. 1–6.

[60] C. Fuchs and T. Hess, "Adapting agile methods to develop solutions for the industrial Internet of Things," in *Proc. 25th Eur. Conf. Inf. Syst. (ECIS)*, May 2017, pp. 1–6.

[61] C. Berger and B. Rumpe, "Supporting agile change management by scenario-based regression simulation," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 504–509, Jun. 2010.

[62] J. Jabeen, "Incorporating artificial intelligence technique into DSDM," in *Proc. Asia–Pacific World Congr. Comput. Sci. Eng.*, Nov. 2014, pp. 1–8.

[63] Y. Y. Jusoh, S. Abdullah, I. M. Ali, M. H. M. Noh, M. H. Mazlan, C. S. Bouh, and T. Z. Sheng, "Adoption of agile software methodology among the SMEs developing an IoT applications," in *Proc. 6th Int. Conf. Res. Innov. Inf. Syst. (ICRIIS)*, Dec. 2019, pp. 1–6.

[64] K. Matsuo, T. Kurita, and L. Barolli, "A new system for management of IoT sensors considering agile-Kanban," in *Proc. Workshops Int. Conf. Adv. Inf. Netw. Appl. (WAINA)*, 2019, pp. 604–611.

[65] S. Jain and S. S. Jain, "Development of intelligent transportation system and its applications for an urban corridor during COVID-19," *J. Inst. Eng. B*, vol. 102, no. 6, pp. 1191–1200, Dec. 2021.

[66] D. Dimitrov and I. Petrova, "Strategic planning and development of transport infrastructures based on agile methodology," *IOP Conf. Mater. Sci. Eng.*, vol. 664, Oct. 2019, Art. no. 012033.

[67] B. Jachimczyk, A. Lubowicki, R. Ogielski, and M. W. B. Kulesza, "The agile-based monitoring and management system for dairy supply chain—The user driven design approach," *Przedsiebiorczosc Zarzadzanie*, vol. 18, pp. 395–412, Nov. 2017.

[68] K. K. Patel, P. G. Scholar, and C. Salazar, "Internet of Things-IoT: Definition, characteristics, architecture, enabling technologies, application and future challenges," *Int. J. Eng. Sci. Comput.*, vol. 6, no. 5, pp. 1–10, May 2016.

[69] P. S. Seemma, S. Nandhini, and M. Sowmiya, "Overview of cyber security," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 7, no. 11, pp. 125–128, 2018.

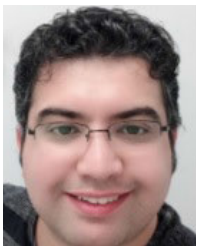
[70] S. Kesh and S. Jane, "Applying agile methodologies to IT security," *Issues Inf. Syst.*, vol. 7, no. 2, pp. 1–4, 2006.

[71] B. A. Jerom and S. P. Rajamohana, "A survey on comparative analysis of agile software development methodologies," *Int. J. Softw. Comput. Sci. Eng.*, vol. 5, no. 1, pp. 10–22, 2020.
[72] S. Shaikh and S. Abro, "Comparison of traditional and agile software development methodology: A short survey," *Int. J. Comput. Syst. Softw. Eng.*, vol. 5, no. 2, pp. 1–14, 2019.
[73] S. A. ALMandhari, E. S. M. H, M. Aissa, and M. A. K, "A comparative study on traditional vs agile systems development methods," *Int. J. Latest Trends Eng. Technol.*, Vol. 15, no. 3, pp. 012–015, 2020.
[74] M. Hneif and S. H. Ow, "Review of agile methodologies in software development," *Int. J. Res. Rev. Appl. Sci.*, vol. 1, no. 1, pp. 1–8, 2009.
[75] E. Alon. (2017). *Scrum*. Accessed: Oct. 28, 2021. [Online]. Available: https://edelalon.com/blog/2017/09/scrum/
[76] M. Fowler and J. Highsmith, "The agile manifesto," *Softw. Develop.*, vol. 9, no. 8, pp. 28–35, 2001.
[77] A. M. M. Hamed and H. Abushama, "Popular agile approaches in software development: Review and analysis," in *Proc. Int. Conf. Comput., Electr. Electron. Eng. (ICCEEE)*, Aug. 2013, pp. 160–166.
[78] S. W. Ambler, "Agile software development at scale," in *Proc. IFIP Central East Eur. Conf. Softw. Eng. Techn.* Berlin, Germany: Springer, 2007, pp. 1–12.
[79] T. Kurita, K. Matsuo, and L. Barolli, "A wheelchair management system using IoT sensors and agile-Kanban," in *Proc. Int. Conf. Intell. Netw. Collaborative Syst.*, Cham, Switzerland: Springer, 2019, pp. 91–100.
[80] O. Sohaib and K. Khan, "Integrating usability engineering and agile software development: A literature review," in *Proc. Int. Conf. Comput. Design Appl.*, Jun. 2010, p. 32.
[81] T. Heinis, J. Heck, F. Fontana, and M. Meboldt, "Agile development for IoT applications: Lessons learned from a case study on hydrants," in *Fundamentals of Internet of Things for Non-Engineers*. USA: Auerbach Publications, 2019, pp. 305–318.
[82] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 833–859, 2008.
[83] D. Strode, "Agile methods: A comparative analysis," in *Proc. 19th Annu. Conf. Nat. Advisory Committee Comput. Qualifications (NACCQ)*, 2006, pp. 257–264.
[84] K. Hiwarkar, A. Doshi, R. Chinta, and R. Manjula, "Comparative analysis of agile software development methodologies—A review," *Int. J. Eng. Res. Appl.*, vol. 6, no. 3, pp. 80–85, 2016.
[85] P. Waddell and A. Borning, "A case study in digital government: Developing and applying UrbanSim, a system for simulating urban land use, transportation, and environmental impacts," *Social Sci. Comput. Rev.*, vol. 22, no. 1, pp. 37–51, Feb. 2004.
[86] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?" *Commun. ACM*, vol. 49, no. 10, pp. 41–46, Oct. 2006.
[87] W. Y. W. Hongwei, "Application of GIS to path choice in agile transportation problem," *Comput. Eng. Appl.*, vol. 9, 2002.
[88] P. Abrahamsson, N. Oza, and M. T. Siponen, "Agile software development methods: A comparative review," in *Agile Software Development*. Finland, 2010, pp. 31–59.
[89] K. Petersen and C. Wohlin, "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Softw. Eng.*, vol. 15, no. 6, pp. 654–693, Dec. 2010.
[90] M. Wadhwa and N. Sharma, "Review of agile software development methodologies," *Adv. Comput. Sci. Inf. Technol.*, vol. 2, no. 4, pp. 370–374, 2015.
[91] M. Brhel, H. Meth, A. Maedche, and K. Werder, "Exploring principles of user-centered agile software development: A literature review," *Inf. Softw. Technol.*, vol. 61, pp. 163–181, May 2015.
[92] M. Poppendieck and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. Reading, MA, USA: Addison-Wesley, 2003.
[93] 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, Oct. 2005.
[94] A. Cockburn and J. Highsmith, "Agile software development, the people factor," *Computer*, vol. 34, no. 11, pp. 131–133, 2001.
[95] E. Hossain, M. A. Babar, and J. Verner, "Towards a framework for using agile approaches in global software development," in *Proc. Int. Conf. Product-Focused Softw. Process Improvement*. Berlin, Germany: Springer, 2009, pp. 126–140.
[96] A. Ahmed, S. Ahmad, N. Ehsan, E. Mirza, and S. Z. Sarwar, "Agile software development: Impact on productivity and quality," in *Proc. IEEE Int. Conf. Manage. Innov. Technol.*, Jun. 2010, pp. 287–291.
[97] V. Szalvay, "An introduction to agile software development," *Danube Technol.*, vol. 3, pp. 688–888, 2004.
[98] A. Felfernig, S. P. Erdeniz, M. Jeran, A. Akcay, P. Azzoni, M. Maiero, and C. Doukas, "Recommendation technologies for IoT edge devices," *Proc. Comput. Sci.*, vol. 110, pp. 504–509, Jan. 2017.
[99] D. Pandit, S. Chowdary, P. S. R. Patnaik, B. Shaharkar, and A. Surde, "Agile methodology for IoT application development and bus. Improvisation," in *Smart Trends in Computing and Communications*. Singapore: Springer, 2022, pp. 601–608.
[100] C. Tam, E. J. D. C. Moura, T. Oliveira, and J. Varajão, "The factors influencing the success of on-going agile software development projects," *Int. J. Project Manage.*, vol. 38, no. 3, pp. 165–176, Apr. 2020.
[101] T. Sato, "Agile program management for chemical engineering and the IoT," *J. Chem. Eng. Jpn.*, vol. 51, no. 9, pp. 826–829, 2018.
[102] J. O. Coplien and G. Bjørnvig, *Lean Architecture: For Agile Software Development*. Hoboken, NJ, USA: Wiley, 2011.
[103] R. M. Barrios, D. Schippers, C. Heiden, and G. Pappas, "A cybersecurity strategy for industry 4.0," *Proc. SPIE*, vol. 11009, May 2019, Art. no. 110090D.
[104] S. Range, "Using agile development methods to enable a threat-based security operations center," Ph.D. dissertation, Dept. Utica College, USA, 2018.
[105] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2001.
[106] S. Zhong, C. Liping, and C. Tian-en, "Agile planning and development methods," in *Proc. 3rd Int. Conf. Comput. Res. Develop.*, Mar. 2011, pp. 488–491.
[107] S. Jeon, M. Han, E. Lee, and K. Lee, "Quality attribute driven agile development," in *Proc. 9th Int. Conf. Softw. Eng. Res., Manage. Appl.*, Aug. 2011, pp. 203–210.
[108] M. Hirsch, "Moving from a plan driven culture to agile development," in *Proc. 27th Int. Conf. Softw. Eng. (ICSE)*, 2005, p. 38.
[109] A. Jøsang, M. Ødegaard, and E. Oftedal, "Cybersecurity through secure software development," in *Proc. IFIP World Conference on Information Security Education*, pp. 53–63. Springer, Cham, 2015.
[110] M. Dark, M. Bishop, R. Linger, and L. Goldrich, "Realism in teaching cybersecurity research: The agile research process," in *Proc. IFIP World Conf. Inf. Secur. Educ.* Cham, Switzerland: Springer, 2015, pp. 3–14.
[111] J. Kinyua, "Cybersecurity in the software development life cycle," in *Cybersecurity for Information Professionals*. USA: Auerbach Publications, 2020, pp. 265–290.
[112] C. Woody and W. Hayes, "Agile and cybersecurity effective risk management is the key," Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. DM19-0441, 2019.
[113] R. Linger, L. Goldrich, M. Bishop, and M. Dark, "Agile research for cybersecurity: Creating authoritative, actionable knowledge when speed matters," in *Proc. 50th Hawaii Int. Conf. Syst. Sci.*, 2017, pp. 1–10.
[114] K. Ikeda, A. Marshall, and D. Zaharchuk, "Agility, skills and cybersecurity: Critical drivers of competitiveness in times of economic uncertainty," in *Strategy & Leadership*. U.K., 2019.
[115] K. D. Willett, R. Dove, and M. Blackburn, "Adaptive knowledge encoding for agile cybersecurity operations," in *Proc. INCOSE Int. Symp.*, vol. 25, no. 1, 2015, pp. 770–792.
[116] H. Gonzalez, R. Llamas-Contreras, and O. Montano-Rivas, "When software engineering meets cybersecurity at the classroom," in *Proc. 7th Int. Conf. Softw. Eng. Res. Innov. (CONISOFT)*, Oct. 2019, pp. 49–54.

YAHYA M. TASHTOUSH received the B.Sc. and M.Sc. degrees in electrical engineering from the Jordan University of Science and Technology, Irbid, Jordan, in 1995 and 1999, respectively, and the Ph.D. degree in computer engineering from the University of Alabama in Huntsville and the University of Alabama at Birmingham, AL, USA, in 2006. He is currently an Associate Professor at the College of Computer and Information Technology, Jordan University of Science and Technology, Irbid, Jordan. His current research interests include software engineering, cybersecurity, artificial intelligence, robotics, and wireless sensor networks.

DIRAR A. DARWEESH received the B.Sc. degree from Al al-Bayt University, Jordan, and the M.S. degree in computer science from the Jordan University of Science and Technology, Jordan. He has worked as a part-time Lecturer with the Jordan University of Science and Technology. His research interests include cybersecurity, the IoT, data mining, artificial intelligence, and cloud computing.

GHAITH HUSARI received the Ph.D. degree from the University of North Carolina at Charlotte, in 2019. He is currently an Assistant Professor with the Department of Computer Science, East Tennessee State University. His research interests include cybersecurity and privacy, big data analytics for cyber threat intelligence, and security analytics and automation.



OMAR A. DARWISH received the M.S. degree from the Jordan University of Science and Technology, Jordan, and the Ph.D. degree in computer science from Western Michigan University, USA. He has worked as a Visiting Assistant Professor with the West Virginia University Institute of Technology, a Software Engineer with MathWorks, and a Programmer with the Nuqul Group. He is currently an Assistant Professor with the Department of Information Security and Applied Computing, Eastern Michigan University. His research interests include cybersecurity, the IoT, and machine learning.

YOUSEF DARWISH received the M.S. degree from the Jordan University of Science and Technology, Jordan. His research interests include software engineering and data mining.

LUAI BANI ISSA received the M.S. degree from the Jordan University of Science and Technology, Jordan. His research interests include software engineering and data mining.

HUTHAIFA I. ASHQAR received the B.Sc. degree (Hons.) in civil engineering from An-Najah National University, Nablus, Palestine, in 2013, the M.Sc. degree in road infrastructure from the University of Minho, Braga, Portugal, in 2015, and the Ph.D. degree in civil engineering from Virginia Tech, VA, USA, in 2018. He also received two graduate certificates in data science and economic development from Virginia Tech, in 2018 and 2021, respectively. He is currently a Senior Transportation Engineer at Precision Systems Inc. in DC and an Adjunct Professor at the University of Maryland Baltimore County. He has authored/coauthored one book chapter and over 30 refereed publications in the areas of innovative mobility, ITS, advanced transportation and energy technologies, AI, and data analytics. His experience includes being a Technical Advisor for programs with over $50 million value in advanced transportation and energy technologies in the U.S. DOE's ARPA-E.