Review on Characteristics of Software Development Frameworks to Reduce Critical Systems Failures

مراجعة خصائص أطر تطوير البرمجيات للحد من حالات فشل الأنظمة الحرجة

اديب احمد جازم محمد ناصر

Adeeb Ahmed Gazem [1]

عبد العزيز احمد ثوابه

Abdulaziz Ahmed Thawaba[2]

https://doi.org/10.54582/TSJ.2.2.94

(1) **Faculty of IT&CS University of Saba Region, Marib, Yemen**
Email: adeahmedg@gmail.com

(2)**Faculty of IT&CS University of Saba Region, Marib, Yemen**
Email: azizth@usr.ac

الملخص :

يعتبر النظام الذي ستكون له تداعيات كبيرة وخطيرة في حالة حدوث فشل أو عطل نظامًا حرجاً. ولذلك، يقوم مطورو البرامج عادةً بتطبيق معايير صارمة واختبارات وامتثال تنظيمي أثناء تطوير الأنظمة الحرجة وتشغيلها وصيانتها لضمان موثوقيتها وأمنها ومرونتها بسبب المخاطر العالية التي تنطوي عليها. يحاول مطورو البرامج والمؤسسات الكبيرة تطبيق أفضل المنهجيات لتطوير الأنظمة الحرجة. لقد اكتسب إطار عمل PMBOK ومنهجيات Agile قبولًا واسعًا من قبل مطوري البرامج. أصبحت هذه الأطر أكثر شعبية لأنها طبقت أفضل الممارسات وجعلتها مناسبة للعمل في إدارة عمليات تطوير الأنظمة الحرجة. الهدف الرئيسي من البحث هو مراجعة الخصائص المرتبطة بفشل الأنظمة الحرجة وتحليل كيفية الحد من فشل الأنظمة الحرجة من خلال استخدام أطر تطوير البرمجيات المناسبة، ويتم ذلك من خلال مراجعة الأدبيات، حيث سيتم مراجعة الدراسات والأبحاث المنشورة حول فشل الأنظمة الحرجة وخصائصها الحرجة. إلى جانب التركيز على أساليب تطوير البرمجيات الرشيقة (ASDM) مثل Scrum و LeSS وSAFe، فإن هذا التحليل يقارنها أيضًا بأفضل ممارسات وتوصيات إدارة المشاريع من PMBOK..

الكلمات المفتاحية: النظام الحرج (CS)، فشل النظام الحرج، إطار تطوير البرمجيات، الهيئة المعرفية لإدارة المشاريع (PMBOK)، إطار العمل المتطور (SAFe)، سكروم واسع النطاق (LeSS)، سكروم

Abstract

A system that will have major and dire repercussions in the event of a failure or malfunction is considered a critical system. Therefore, Software developers usually apply strict standards, testing, and regulatory compliance while developing, operating, and maintaining critical systems to ensure their reliability, security, and resilience due to the high risks involved. Software developers and large organizations try to apply the best methodologies to develop critical systems. The PMBOK framework and Agile methodologies have gained wide acceptance by software developers. These frameworks became more popular because they applied best practices and made them suitable for working in the management of critical systems development processes. The main objective of the research is to review the characteristics associated with the failure of critical systems and analyze how to reduce the failure of critical systems through the use of appropriate software development frameworks. This is done through a literature review, where published studies and research on the failure of critical systems and their critical characteristics will be reviewed. Along with emphasizing Agile Software Development Methods (ASDM) like Scrum, LeSS, and SAFe, this analysis also contrasts them with PMBOK's best project management practices and recommendations.

Keywords: Critical system (CS), Failure of critical system, Software Development Framework, Project Management Body of Knowledge (PMBOK), Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), Scrum.

1. Introduction

A critical system is any system whose "failure" could endanger human life, the environment in which the system operates, or the organization that maintains the system. Failure in this context refers to any actions that pose a risk to the system. Before computer-based systems, there were other crucial systems like airplanes and missiles. In critical systems, the cost of failure could be more than the cost of the system itself. Indirect costs may be higher than direct failure costs [1]. The structured method for creating software for a system or project is known as the software development process [2]. Planning, implementation, testing, development, and maintenance are the four processes that make up this process due to the great growth of the information technology sector, which came about as a result of the market's expanding business, health, and other sectors. Due to the rising pressure of the need for innovation, new and varied software development models have been employed and developed, which has had an impact on the conventional models of software development management in a more complex way. There are benefits, traits, and drawbacks to each methodology such as Scrum, LeSS, SAFe, and PMBOK. Scrum is a lightweight framework that helps people, teams, and organizations generate value through adaptive solutions to complex problems, and various processes, techniques, and methods can be used within this framework. Scrum's focus on delivering integrated, tested, business-valued features in every iteration results in rapid delivery of results [3]. LeSS is Scrum applied to many teams working together on a single product [4]. Cross-functional teams all serve the same purpose, and the teams decide on the advantage that should be worked on based on the strengths of all the teams striving for continuous integration of features and continuous delivery of shippable products [5]. A collection of organizational and workflow patterns called the Scaled Agile Framework® (SAFe®) is used to deploy agile methods at an enterprise scale. The framework is a body of information that offers organized advice on roles and responsibilities, how to organize and manage the job, and principles to uphold. Over a wide number of agile teams, SAFe encourages alignment, collaboration, and delivery [6]. Contrarily, the PMBOK (Project Management Body of Knowledge) is not a methodology in the strictest sense. It is a collection of project management rules, jargon, and best practices. The PMBOK Guide which is published and updated by the Project Management Institute (PMI). PMBOK provides a clear set of instructions for project management as a whole, rather than one particular project management methodology [7]. The main objective of the research is to review the characteristics associated with the failure of critical systems and analyze how to reduce the failure of critical systems through the use of appropriate software development frameworks. This is done through a literature review, where published studies and research on the failure of critical systems and their critical characteristics will be reviewed. Some of the most famous development and project management frameworks such as Scrum, LeSS, SAFe, and PMBOK will also be reviewed and studied to find out which ones will contribute to developing critical systems and reducing their failure. Safety-critical systems are created to avoid catastrophic failure-related

outcomes, such as human injury or death and environmental harm [8]. A framework that can adapt to changing working conditions is necessary due to IoT technologies, complex client expectations, increased market volatility, the dynamic and frequently unforeseen character of these changes, and the complexity of systems in which work is embedded. The path of future research in the area of developing safety-critical systems is highlighted by an understanding of the key factors that could cause the failure of safety-critical systems and by a review of the present development frameworks [9] the associated risks hamper its implementation and lack a comprehensive overview. In response, the paper proposes a framework of risks in the context of Industry 4.0 that is related to the Triple Bottom Line of sustainability. The framework is developed from a literature review, as well as from 14 in-depth expert interviews. With respect to economic risks, the risks that are associated with high or false investments are outlined, as well as the threatened business models and increased competition from new market entrants. From an ecological perspective, the increased waste and energy consumption, as well as possible ecological risks related to the concept "lot size one", are described. From a social perspective, the job losses, risks associated with organizational transformation, and employee requalification, as well as internal resistance, are among the aspects that are considered. Additionally, risks can be associated with technical risks, e.g., technical integration, information technology (IT.

2. Theoretical Background

Theoretical literature related to critical system failures and software development frameworks will be relied upon. Through a review and study of various well-known development and project management frameworks, the features and advantages of these frameworks will be examined to identify which frameworks are most suitable for developing critical systems and reducing critical system failures.

2.1 Critical Systems

A critical system is one whose "failure" could endanger human life, the environment in which it operates, or the organization that manages it. One of the most significant instances of critical systems is the aviation and healthcare systems. Critical systems can be divided into three groups: (1) Systems deemed "safety-critical" that are those systems in which malfunction might cause grave environmental damage, or human fatalities, or both; (2) Systems that are mission-critical, or whose failure could cause another goal-directed activity to fail; (3) A system classified as "business-critical" that is one whose failure could cause the firm that uses it to fail. Safety-critical systems (SCS) are essential to our daily operations, but any malfunctions in their functioning have the potential to result in severe financial loss, harm to people or the environment, or even fatalities. SCS are utilized in crucial industries like transportation, healthcare, nuclear, and space systems. Any errors in its functioning may lead to serious consequences. Enhancing performance across development phases could prevent more than 59% of SCS

failures [10]. Intelligent robotics, autonomous driving, and robot surgeons all fall under the category of safety-critical technologies, which call for extremely high levels of reliability while in use. Therefore, when developing safety-critical intelligent systems, the issues of safety and reliability must be taken into consideration [11]intelligent robotics and medical surgeon robots, require a stringent dependability while the systems are in operation. Therefore, the safety and reliability issues must be addressed in the development of safety-critical intelligent systems. Nevertheless, the incorporation of the safety/reliability requirements into the system will raise the design complexity apparently. Furthermore, the international safety standards only provide guidelines and lack concrete design methodology and flow. Therefore, developing an effective safety process to assist system engineers in tackling the complexity of the system design and verification, and in the meantime, satisfying the requirements of international safety standard, become an important and valuable research topic. In this study, we will propose a model-based safety-critical system design, analysis and assessment framework which incorporates fault tree-based weak-point analysis, system hardware architecture exploration and safety mechanism effectiveness assessment with model-implemented fault injection. Failure modes and diagnostic coverage analysis (FMEDA.

2.1.1 Failure of Critical Systems

Safety-critical systems are now used in many other fields, including the Internet of Things and medical devices. Such systems, however, are frequently exceedingly complicated and prone to errors. As a result, real-time, safety, and security concerns are frequently only partially taken into account. Failure and change impact analyses must be used to examine the effects to reduce or prevent safety-critical failures [12]. As seen in Figure 1, 85% of failure cases happen during the development stages and result in the causes of the failure emerging during system operation [13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD. For safety-critical systems, safety and reliability are crucial concerns. As a result, monitoring, fault diagnosis, and failure prognosis are crucial to the safe operation of these systems. The main priority for the system in safety-critical systems is isolating an impending defect and then forecasting the future behavior of malfunctioning components. Knowing how much longer the system can operate with an appropriate response is very important. Furthermore, for the system to function at its best, forecast accuracy is essential. Therefore, new techniques are required to guarantee higher system reliability as safety-critical systems become more complex [14]. When developing such safety-critical intelligent automotive systems, reliability and safety issues must be taken into account because they must have strict reliability when the systems are in use. However, adding safety and reliability criteria to the system will significantly

https://doi.org/10.54582/TSJ.2.2.94

increase design complexity [15]safety and reliability issues must be addressed in the development of such safety-critical systems. Nevertheless, the incorporation of safety/reliability requirements into the system will raise the design complexity considerably. Furthermore, the international safety standards only provide guidelines and lack concrete design methodology and flow. Therefore, developing an effective safety process to assist system engineers in tackling the complexity of system design and verification, while also satisfying the requirements of international safety standards, has become an important and valuable research topic. In this study, we propose a safety-oriented system hardware architecture exploration framework, which incorporates fault tree-based vulnerability analysis with safety-oriented system hardware architecture exploration to rapidly discover an efficient solution that complies with the ISO-26262 safety requirements and hardware overhead constraint. A failure mode, effect, and diagnostic analysis (FMEDA. Safety-critical systems (SCS) reliability analyses, such as reliability, safety, performability, etc., have been modeled using a variety of modeling methodologies, including fault trees, failure mode effect analysis (FMEA), reliability block diagrams (RBDs), and unified modeling language (UML). These methods fall short of capturing dynamic activity and only show the static characteristics of a system [16]. Additionally, the international standards for safety only offer guidelines and don't specify a design methodology or flow, so creating a practical safety procedure to help system engineers handle the complexities of systems development, verification, and examination while also meeting the requirements of the international standards for safety has turned into a significant and worthwhile study topic. Late identification of safety and security design flaws can increase costs, add to system complexity, resulting in ineffective mitigation measures, and cause delays in system deployment [17]. The design phase and the development phase are the next most frequent causes of failure, according to Figure. 1 Thawaba et al. [13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD, followed by system analysis.

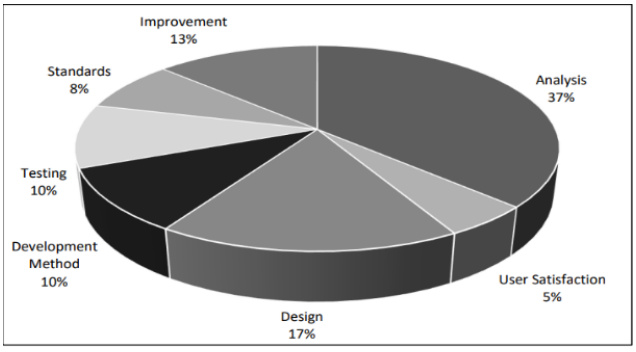

Figure 1. Failures arise in the development stages of SCS[13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD.

Figure 2 displays the SCS failure regions based on information from earlier studies. The development section is where the majority of failures occur. Additionally, faults occur at variable rates during the whole development process, which could result in failures throughout the system's operational stage. The SCS's failure resulted in a humanitarian catastrophe. Therefore, before beginning development, developers should: (1) thoroughly research the SCS; (2) be aware of the repercussions of any errors made; (3) train the team of development; (4) examine related systems; and (5) incorporate SCS specialists in the development team [13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD.

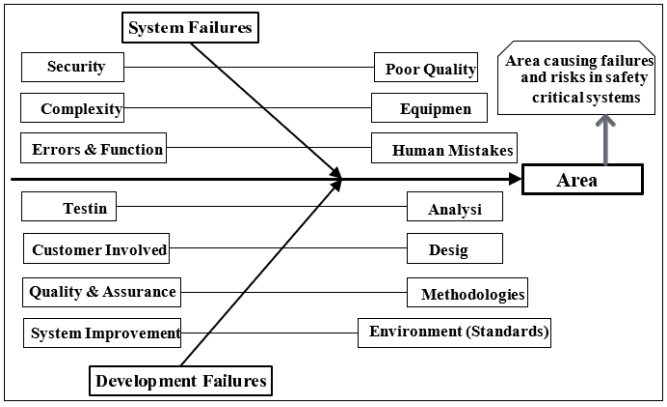

Figure 2. Cause and Effect Diagram for Critical Systems Failure[13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD

2.1.2 Characteristics of Safety Critical Systems

SCS engineers are working very hard to enhance SCS and lessen failure factors. In SCS development, four criteria can be used to minimize failure points. Understanding Failure: when developers are aware of the most frequent reasons for SCS failure, they may come up with strategies to get around them when developing. Choosing Development Practices: deciding on the best SCSD approach based on (the system delivery mechanism, the culture of the developers, and their expertise). Standards "Obtain Quality": programmers need to understand which national and international standards the system should adhere to. SCSD testing is system-dependent and lacks a dedicated test technology. Project management tests and technical tests are the two different categories of tests [13]railway systems, medical devices, nuclear systems, and military weapons. SCS failures could result in losing life or serious injuries. Improving the practices during development phases of SCS can reduce failures up to 40%, thus resulting developers to follows specific development practices and techniques. Developers should improve safety-critical system development (SCSD.

2.2 Reliability Assessment

The primary attribute of quality is reliability, which is monitored by numerous tests throughout development [18]. The increased complexity of producing goods and tiny components has sparked an increased interest in driving reliability as a result of new business requirements and technological advancements. Reliability measures are taken into consideration in numerous industries [19] due to the rise in the quantity and diversity of faults. The difficulty of proving the reliability of SCS systems during development is brought on by the repercussions of the failure of safety-critical systems [20]. By concentrating on the reliability of individual components as well as the overall reliability of the system's constituent parts, system reliability can be attained [21]. Estimating the project's system reliability is a crucial evaluation step when developing contemporary intricate systems, such as reactors for nuclear power. Multiple strategies are employed, including backup, regulation of fault-tolerant control systems, and generalization, to attain the necessary reliability indicators. The main information used to assess system reliability is structure and reliability diagrams, a list of components with reliability indicators (failure rate or failure-free), and reliability targets for systems that are still in development. It is suggested that the system be divided into three tiers to improve the reliability modeling processes [22]:

Component level: The basic building blocks of the system (such as controllers, input and output units).

Operational level: It establishes the specifications for reliability indicators and details the structure of the effect of component health on the functionality of a system's func-

tional process.

At the system level, the structure of the effect of system success on system failures is explained, and estimated final system reliability indicators are produced.

software reliability is a characteristic that the software constantly offers to the service without experiencing any failures. This concept is used in software engineering. There are numerous model-based techniques for determining software reliability using statistical data that have been developed during the past few decades. The data gathered during the testing process is used by the model to evaluate the software's reliability [23]. To increase reliability in terms of software size and use software development procedures to minimize and control faults, evaluation models comprise parameters that are estimated using recent data [24]. Software engineers must do quantitative analysis throughout the entire development process to make an acceptable choice, such as defining a test endpoint, to assure software reliability [25]. Three actions make up software reliability. It began with fault isolation, detection, and correction, and it concluded with measures to improve reliability. The efficiency with which the phases and tasks are carried out serves as a gauge of the project development processes' reliability. When designing SCS, reliability must be measured individually for each task or subtask before being aggregated to estimate the reliability of the overall project..

2.3 Development Frameworks

The methodologies used in software development have changed over time, moving from predictive (such as Waterfall) to incremental and iterative (such as Agile Application Development, Rational Unified Process), to agile (such as Scrum, XP), and finally to large-scale agile (such as SAFe, LeSS). Conventional development techniques, such as the V-Model and Waterfall Model, are usually used to design critical systems because they are unambiguous, simple, and compliant with legal requirements, like providing documentation that aids in traceability. These approaches, however, can be costly and time-consuming, and they might not be appropriate when the needs alter. Conversely, agile approaches offer numerous advantages, like producing high-caliber software quickly and at a reasonable cost, as well as the flexibility to adapt to requirements changes as they arise [26]. Project management methodologies (PMMs) and different agile software development methodologies (ASDMs) are being used in place of traditional software models and procedures for software development projects. Numerous businesses end up selecting just one of these two strategies to undertake software development projects since these approaches struggle to meet all the issues associated with software development projects [27]. Agile methods are a valuable tool for enhancing productivity and quality. They can be applied in the development of safety-critical systems, where quality is the primary focus, despite safety being the primary concern in safety-critical development [28]. Safety-critical businesses are seeing

increased pressure to embrace agile development practices in the software industry to respond to changing needs faster and deliver systems to clients more frequently for evaluation and integration [29]. The most well-known agile software development methods (ASDMs), including Scrum, SAFe, and LeSS, as well as one of the most well-known project management methodologies (PMMs), PMBOK, will be covered in this section..

2.3.1 PMBOK

The extensive collection of accepted industry best practices, conventions, and processes is known as the Project Management Body of Knowledge, or PMBOK for short [30]. The PMBOK is the most widely used benchmark for evaluating project management systems and the most well-known worldwide standard in project management [31]. Businesses see the PMBOK as helpful since it helps them prevent project failures, standardize procedures throughout departments, and adapt procedures to specific requirements. Because practitioners continuously expand the body of knowledge by discovering best practices or new techniques, it must be updated and disseminated often. The PMBOK is an industry framework that integrates best practices in project management; it is not a technique, rather it is an endeavor sponsored by the Project Management Institute (PMI). It offers a thorough explanation of numerous incredibly helpful tools and methods to assist you in managing your project more effectively [32]. Although it is compatible with more recent approaches like Agile, it is sometimes linked to the waterfall methodology, which linearly arranges project stages. The PMBOK's processes can be tailored to suit a variety of project management contexts, which is why the PMI does not endorse any specific methodology. Rather, managers select the procedures that work best for their organizations, projects, and teams. The practical framework and standards that PMBOK offers, which are derived from years of excellent project management practices by thousands of project managers, are its strengths and advantages. They also represent a process-oriented approach. Additionally, it may be applied to project management in a variety of industries, where inputs, tools, procedures, and outputs fully describe each process. Also, it is quite challenging to define and accomplish success in information systems project management because of the abundance and diversity of engaged stakeholders and variables that project managers and their teams must take into account. In this context, project management standards and guidelines are helpful since they offer ideas, procedures, and methods on various connected fields of knowledge (e.g., quality, risk, cost, etc.). They do not, however, specifically state what must be done to manage a project's success (such as the formal evaluation of success) [33]. The PMBOK Guide's shortcomings were noted by Simon Buehring as follows: (i) the project management team members' lack of clearly defined duties; (ii) the excessively intricate and thorough explanations of a few of the components; (iii) the PMBOK Guide is unavoidably written from a North American viewpoint and isn't necessarily as adaptable to other cultural contexts [32].

https://doi.org/10.54582/TSJ.2.2.94

2.3.2 Agile Software Development Methods (ASDM)

A flexible and iterative approach to software development, the Agile Software Development Methods (ASDM) place a strong emphasis on cooperation, adaptation, and continual improvement. Its cornerstone is the Agile Manifesto, which prioritizes customer collaboration, software development, human interaction, and change adaptation [34]Agile is a software development framework, which has developed software by using 1-4 week iterations, which will produce better quality software as per customer requirements. Agile methodology mostly deals with the frequent changing requirement in the software project. Instead of traditional method, it is always beneficial to use Agile as a stakeholder because traditional software methodologies are less prone to changes. In this research paper, authors have studied the effect of agile approach in the software development process in terms of quality, business worth and architectural framework.",”container-title”:”2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC.

i. Scrum Methodology

The primary goal of the Scrum approach, which was created in 1993 by Jeffrey Sutherland at the Easel Corporation as a framework for project management, is to manage complicated projects with quality in short amounts of time, particularly for software development. Scrum is an innovative product and service development methodology that is built on an agile approach and a limited set of fundamental beliefs, practices, and principles, collectively referred to as the Scrum framework. The Scrum framework comprises three primary principles—transparency, inspection, and adaptation—that give structure and overviews. It also allows for ceremonies (Events), roles, and artifacts [35]. There are three key roles in Scrum: (i) The Product Owner is in charge of outlining and organizing the features and functionality that are necessary, works closely with the Scrum Master and development team, responds promptly to inquiries, represents stakeholders, and makes sure the project adds value for the company; (ii) The Scrum Master serves as a coach, leads the Scrum process, and aids the team and organization in achieving high-level Scrum performance. They also help all participants comprehend and accept the practices, values, and principles of Scrum. It also supports the team in resolving issues and enhancing its application of Scrum, as well as helping with challenging change management that may arise during Scrum implementation. It shields the group from outside influence and gets rid of roadblocks that get in the way of teamwork. Not a manager, but a leader, is what the Scrum Master does; (iii) A development team consists of a variety of cross-functional individuals who work together to design, develop, and test a desired product. To accomplish the objective that the product owner has set, the team self-organizes. A team typically consists of five to nine members, all of whom need to be highly skilled to produce software of the highest caliber. It is possible to build multiple Scrum Teams for development projects that call for larger teams. Since they are all focused on the same product, they must have the same product owner, product backlog, and goal [36]. The following show how the

https://doi.org/10.54582/TSJ.2.2.94

Scrum process is implemented [37]:

- Determine the Product Backlog: The product owner prepares the backlogs that will comprise all the requirements of the project and collects them into the product backlog. The features that will be developed are decided by the scrum master based on the needs and priorities.
- Sprint planning involves the Scrum Team discussing and assessing the features and items in the product backlog that the Product Owner has requested, calculating the number of hours needed for every feature, estimating the cost, and allocating responsibilities to team members. Tasks that need to be completed inside the Sprint are compiled into a list called the Sprint Backlog. Any modifications made during development ought to wait until the following Sprint.
- Daily Stand-Up Meetings: During these sessions, the effectiveness of the team members working on the features under development is assessed. Based on the amount of work accomplished, the sprint completion time is evaluated. Plans for developmental tasks are compared against completed and ongoing tasks to ensure progress is being made. These development activities sometimes referred to as the sprint board, include the storyboard associated with that particular sprint. The storyboard includes issues found in the development and testing phases. A visual display of the project's progression is the burndown chart.
- Sprint Review: The team gathers input and shows the product owner and customer the features that have been produced throughout this process.
- Sprint Retrospective: This procedure entails identifying and talking about activities that are too difficult to continue into the following sprint. We also talk about tasks that are finished and can carry over into the following sprint. The development process is enhanced and the team's performance is assessed during this phase. The Scrum process stages are shown in Figure 3.

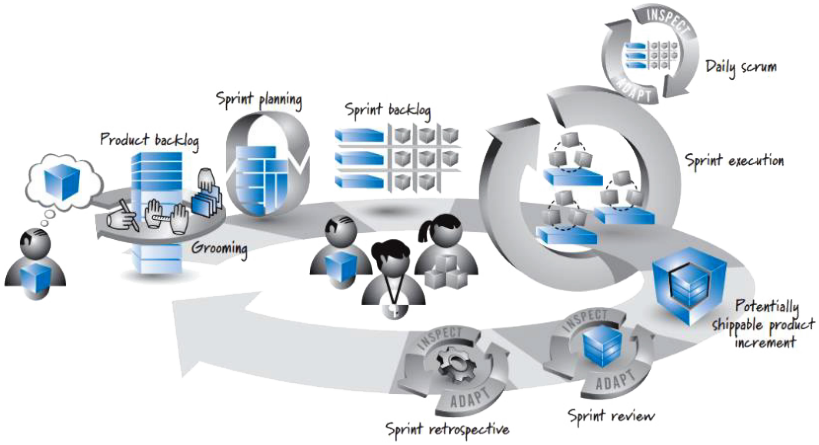

Figure 3. SCRUM Framework [3]

Changes in requirements get less expensive and easier to adopt since Scrum is an iterative process that involves close engagement with the Product Owner. The primary benefits of Scrum are its adaptability to change flexibility, and self-organization. Nevertheless, drawbacks include the fact that risk management procedures are not always formal, tasks can occasionally be too complex for the team to handle, and the development team is ultimately responsible for all work [35]. The lack of documentation and the need for experienced developers due to the size of small teams can be seen as weaknesses of Scrum. However, the strength of Scrum is that it enables quick reactions and delivers products in short cycles and that it is a methodology that can be combined with different methodologies [38]. .

ii. Scaled Agile Framework (SAFe)

SAFe is a collection of guidelines and procedures that enable the implementation of an agile workflow across the whole company. SAFe specifically highlights 10 principles, a Lean-Agile mentality, and four essential values (alignment, built-in quality, transparency, and program execution) [39]. The framework includes an implementation roadmap to help firms through the change and is built on fundamental values and principles [40]. To improve the transparency and clarity of program performance, SAFe offers a variety of tools for a range of uses, including feature progress charts, burn-up charts, program Kanban boards, and continuous flow diagrams. SAFe offers twenty different kinds of metrics at every level of the framework, such as the enterprise-level business agility self-assessment, the value stream KPI, and the portfolio-level lean portfolio metrics. With SAFe, every organization can modify the framework to suit its own set of requirements. SAFe provides four innovative configurations to support the entire range of solutions, from simple systems that can be designed and deployed by a small team to sophisticated systems that take hundreds or even thousands of people to develop. SAFe comes in a variety of configurations. Guidelines exist for four distinct organizational levels in the most comprehensive configuration: team, program, value stream/solution, and portfolio. A few roles and actions are illustrated in Figure 4 [41]..

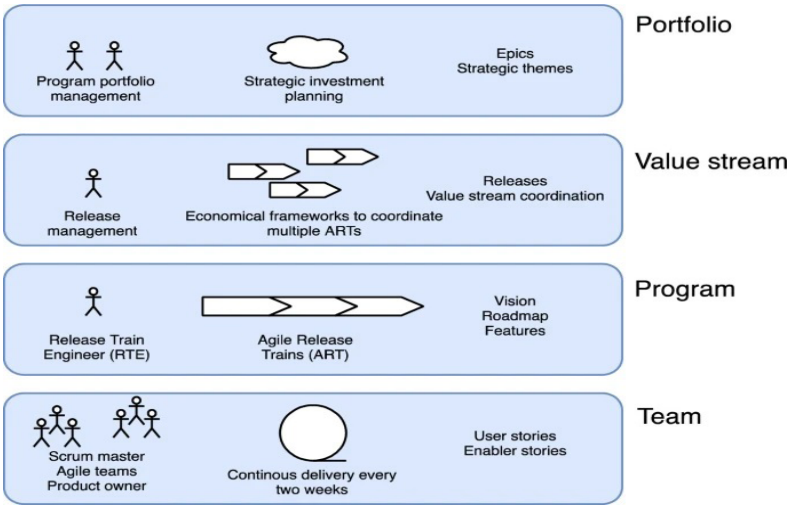

Figure 4. Activities and Roles Covered in Every SAFe Level

At the team level, at least every two weeks, the product owner, agile teams, and the scrum master operate and provide working systems. User and enabler stories serve as the main foundation for the development process. An Agile Release Train (ART) is responsible for coordinating the agile teams at the program level. Several designs, such as "one ART developing one value stream," "one ART developing multiple value streams," and "several ARTs developing one large value stream," can be found, based on how many participants are in the ARTs [42]. There are usually many dependencies amongst ARTs when they are supplying a single, big value stream. Organizations that need more responsibilities to integrate the operations of interdependent complex systems are categorized as value stream level. The process of developing solutions that create continuous value for clients and can either directly benefit the client or assist internal operations is known as the value stream definition. Teams are organized into ARTs, to create durable structures of organization made up of teams that are agile, important stakeholders, and additional resources after the value streams have been identified. At the value stream level, release management positions collaborate in conjunction with economical frameworks for coordinating various ARTs and value streams. Through program portfolio management, the Portfolio level seeks to coordinate the value streams across the lower levels to meet the portfolio's financial and business goals of the portfolio as well as the firm's overall business goals of the firm. This level includes what are known as "epics," which are endeavors that cut across all organizational levels, from the upper levels' visions to the lower levels' actual development projects [43].

To deploy SAFe and embrace agility throughout the organization, executives must provide their whole support. SAFe focuses on simultaneous process, role, and tool adaptation throughout the entire organization rather than team procedures or workflow [39]. The main success factors of SAFe are structured planning, transparency, work visibility, communication, coordination between teams, management, and stakeholder engagement; however, team members' main challenges are rigidity, lack of innovation, dependency between teams, and lack of a design process [44]. Among SAFe's shortcomings are its excessive prescriptiveness, its rigid structure, and its top-down-driven methodology [45].

iii. Large-Scale Scrum (LeSS)

LeSS is Scrum implemented for multiple teams collaborating on the same project. Craig Larman et al. created the Large-Scale Scrum (LeSS) framework, as shown in Figure 2.5. The goal of cross-functional teams is to continuously integrate features and provide shippable products. Based on the strengths of each team, the teams determine which advantage should be prioritized. LeSS 2 to 8 Teams and LeSS Huge 8+ Teams are the two frameworks used in Large-Scale Scrum. LeSS comprises ten principles, practices, and instruments. Small-batch work is continuously integrated among the different teams, allowing frequent product delivery for users. These quick cycles also continuously give feedback that enables planning and maximizing the agility of the organization [5]. The change at LeSS is ongoing through experimentation and improvement; this approach reduces the need for traditional project/program management and enables a considerably shorter deployment procedure. The structure proposed in LeSS is also considered suitable for ensuring high follow-up of operations, although it becomes a little reactive to changes that may arise in requirements [6]. LeSS places the consumer at the center and works exclusively with the product, not with programs or initiatives. By establishing parameters only for product management, and organizational structure, and collaborating with various teams within a single sprint rather than imposing a set of strict rules, LeSS allows the executor a great deal of flexibility [46]. Its shortcomings, according to Ömer Uluda, are that it is a drastically agile methodology that could be challenging to implement in sizable traditional businesses and that it also necessitates an ideal agile setup and skilled agile software developers [45]..

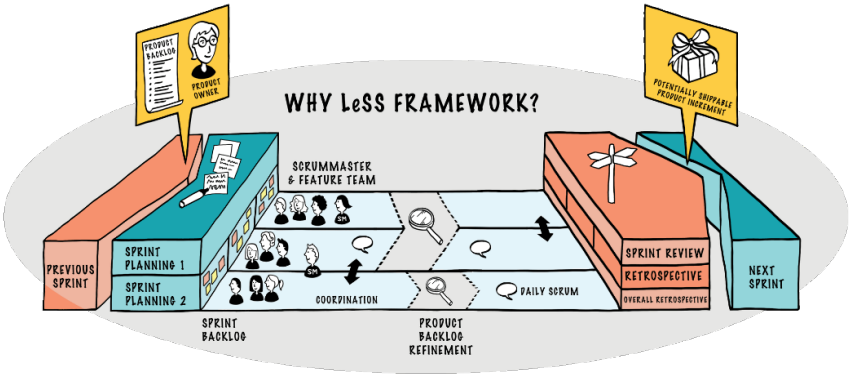

Figure 5. The Smaller LeSS Framework [4].

The smaller LeSS architecture is meant for one (and only one) Product Owner to oversee one Product Backlog that teams work through in one Sprint to optimize the product as a whole. The LeSS framework shares many of the same components as one-team Scrum [5]:

- Positions: Two to eight Teams, one Product Owner, and one to three Teams each have a Scrum Master.
- Artifacts: One Product Backlog, one possibly shippable product increase, and a distinct Sprint Backlog for every Team.
- Events: All teams participate in a single, common sprint that culminates in a single, potentially shippable product increment.
- Guidelines & Rules: Guidelines for an empirical process control and whole-product focus framework with just enough scale.

3. Research Methodology

The main objective of this research is to review the characteristics associated with critical system failures and analyze how using appropriate software development frameworks can reduce critical system failures. This is achieved through a review of the literature, including published studies and research on the failure of critical systems and their critical characteristics. In addition, the most popular agile software development methods (ASDMs), such as Scrum, SAFe, and LeSS, as well as one of the most popular project management methodologies (PMMs), PMBOK, will be examined and studied to identify frameworks that contribute to the development of critical systems and help mitigate their failures.

4. Discussions

This section compares the development frameworks discussed and discusses whether there is a framework that is most suitable for developing critical systems taking into account their critical characteristics. Moreover, some points and practices that contribute to reducing critical system failures will be discussed.

4.1 Comparison of Characteristics of ASDM Methodologies

Software development projects that demand flexibility, quick delivery, and the capacity to adapt to changing requirements frequently employ ASDM. ASDM's popularity has grown as a result of its capacity to meet changing customer needs and provide value promptly. It may be necessary to take into account several aspects when choosing the right ASDM development framework, including (1) corporate culture and team experience; (2) the best way to deliver the system in its entirety or parts; (3) the system's size and kind; and (4) time and cost [15]. To shed further light on the distinctions between the three ASDM models covered in this study, a comparison of their principles, practices, tools, metrics, roles, and artifacts will be conducted. Table 1 describes the ASDM comparison.

Table 1 Comparison between ASDM methodologies [3], [39], [47], [48]

	SAFe	LeSS	Scrum
Principles	Take an economic view, (2) Apply (1) systems thinking, (3) Assume variability; preserve options, (4) Build incrementally with fast, integrated learning cycles, (5) Base milestones on objective evaluation of working systems, (6) Visualize and limit Work in Process (WIP), reduce batch size, and manage queue length, (7) Apply cadence; synchronize with cross-domain planning, (8) Unlock the intrinsic motivation of knowledge workers, (9) Decentralize decision-making, (10) Organize around value	Large-Scale Scrum (1) is Scrum, (2) More with LeSS, (3) Lean Thinking, (4) Systems Thinking, (5) Empirical Process Control, (6) Transparency, (7) Continuous Improvement Towards Perfection, (8) Customer Centric, (9) Whole Product Focus, (10) Queuing Theory	Core Values: Commitment, Focus, Openness, Respect, and Courage Scrum is founded on: empiricism and lean thinking Principles: transparency, Inspection, Adaptation
Practices	Build solutions component with high functioning ARTs; Build integrate with a solutions train; Capture and refine systems specification in solution intent; Apply multiply planning horizons; Architect for scale, modularity, reusability, and serviceability	Sprint planning, Sprint Review, Retrospective, Overall retrospective, Daily Scrum, Coordination and Integration (Just Talk), Communicate in Code	Sprint planning, Sprint Review, Sprint Retrospective, Daily Scrum, Creating a (Product Backlog, and Increment
Tools	Progress feature chart, program Kanban boards, burn-up charts, a continuous flow diagram	Free open-source (FOSS) tools, Free Web 2.0 information tools	Not reported explicitly
Metrics	Lean Enterprise Assessment, Lean Portfolio Metrics, Enterprise Balanced Scorecard, Lean Portfolio Management Self-Assessment, Value Stream Key Performance Indicators	Not reported explicitly	Team Velocity, Customer Satisfaction, Defect Count
Roles	Agile team, Scrum master, Product owner, Product manager, System architect, Release train engineer, Business owner, Solution manager, Solution architect, Solution train engineer, Epic owner, Enterprise Architect	Teams, (Area) product owner, Scrum master, Managers	Developers, Product owner, Scrum master
Artifacts	Story, Enabler story, Iteration goals, Team backlog, Feature, Enabler feature, Program increment objectives, Program Board, Program backlog, Solution intent	Area) product backlog,) Sprint backlog, potentially shippable product increment	Product backlog, Sprint backlog, Increment

4.2 Comparison of ASDM Characteristics with PMBOK

It is noteworthy that ASDM and PMBOK are complementary and can be utilized in tandem on specific projects. To meet the unique requirements of their projects,

some businesses may choose to use a hybrid strategy that incorporates components of the two frameworks. A number of variables, including project complexity, requirements volatility, and corporate culture, influence the decision between ASDM and PMBOK. There are two distinct approaches to project management and software development: the PMBOK framework and the ASDM framework. Table 2 presents a few salient features and areas of overlap between the two. Refers to the methodology used for system development, such as Iterative or Linear. Ability to implement large projects: Capability of the methodology to successfully handle and manage large-scale projects. Transparency: The ability of the methodology to present information clearly and comprehensibly to all team members and stakeholders. Team size: Number of individuals involved in the project team. Geographically distributed: Indicates whether the team is geographically dispersed or not. Expandability: The ability of the methodology to handle project expansion and increased scale without negatively impacting performance or quality. Complexity: Expected level of complexity in implementing the methodology. Coverage: The extent to which the methodology encompasses and applies to every facet of the project. Customer involvement: Degree of customer participation and continuous communication with the team during the development process. Flexibility: The ability of the methodology to adapt to changes and evolving project requirements. Time to market: The amount of time needed to finish the project and launch it in the market. Detailing and support level: Level of detail and support available from the methodology and the team during and after project implementation. Cost: Cost of implementing the methodology and providing necessary resources for the project. Accommodate changes: Ease with which the methodology can accommodate changes and modifications during the development process. Continuous improvement: Ability of the methodology to enhance processes, products, and performance based on previous experiences. Ease of use: Ease of using and comprehending the methodology by team members. Learning ability: Capability of the methodology to absorb and incorporate learning from past mistakes and improvements. Waste elimination: Focus of the methodology on eliminating unnecessary processes or waste in the development process. Application area: The domain in which the methodology can be applied, such as software development, transportation systems, etc. Documentation: Documentation level of all details of development processes. Difficulty transformation: Expected complexity and potential challenges during the transformation of the methodology from one development system to another. Possibility of merging with other methodologies: Capability of the methodology to integrate and collaborate with other development methodologies.

Table 2. Comparing the Characteristics of ASDM and PMBOK Frameworks

method / Parameters	PMBOK	LeSS	Scrum	SAFe
Development method	Linear	iterative	iterative	iterative
Ability to implement large projects	High	High	Low	High
Transparency	Low	High	High	High
Team size	High	Moderate	Low	High
Geographically distributed	------	Moderate	Moderate	High
Expandability	-----	Moderate	Low	High
complexity	High	Moderate	Low	High
Coverage	High	Moderate	Moderate	High
Customer Involvement	Low	High	High	High
Flexibility	Low	Moderate Or low	High	Low Moderate
Time to market	Low	Moderate	High	Moderate
Detailing and support level	High	Moderate	Moderate	High
Cost	High	Moderate	Low	High
Accommodate changes	Low	Low	High	Low
Continuous improvement	Low	High	High	Low
Easy to use	Low	Low	High	Moderate
Learning ability		High	High	Low
Application area	Software and non-software projects	Software	Software and non-software projects	Software
Documentation	High	Low	Low	Low
Difficulty transformation		High	Low	High
Possibility of merging with other methodologies			High	

المجلد(7) العدد(1) يونيو 2024م
https://doi.org/10.54582/TSJ.2.2.94

4.3 Development Framework and Critical System Failure

By reviewing the causes of failure of critical systems, we found that 85% of failure cases happen during the development stages and result in the causes of the failure emerging during system operation. This led us to review the current development frameworks used to develop critical systems. Although developers are still using traditional approaches such as the V model and PMBOK, there has become a tendency from the industry market to move towards agile methodologies for the development of critical systems for reasons such as (high quality, finding out mistakes early, speed of product access to the market, customer satisfaction and other advantages provided by ASDM). Although there are still risks, developers continue to adapt ASDM to suit critical systems.

Not every ASDM is suitable for every critical system development project, and developers' culture, experience, geographical distribution and their acceptance of this transformation must be taken into account. All members of the organization must be made aware of the importance of shifting from traditional methodologies to ASDM, as well as providing them with appropriate training and providing the organization with people who have appropriate experience to ensure the success of this transformation and enhance their awareness of all challenges that they will face during this transformation. Because critical systems are involved in all vital software industries such as (space stations, nuclear power plants, military weapons, aviation systems, train systems, car systems ... etc.), their importance has emerged and has increased in complexity. Because of the severe consequences of its failure, such as loss of life, injury, or severe damage to the environment, the cost of failure is likely to exceed the cost of the system itself. In addition to the direct costs of failures such as (loss of income when the system stops), there are indirect costs resulting from system failure that may be greater than the direct costs, such as (the cost of verifying the cause of the problem, compensation costs, legal costs associated with compensation claims, redesign costs, changing other systems that may be subject to the same type of failure, loss of reputation of the institution, loss of trust in its services), therefore critical characteristics emerge such as (reliability, availability, safety, security, and maintainability) for the development of such systems. To ensure the achievement of these characteristics, the importance and complexity of tests increases. Each development methodology has characteristics, advantages, and shortcomings; see Table 2. To avoid failure during the development of critical systems, The organization must:

- Study the critical system concerned with the development carefully to know the critical characteristics of the system (identify difficulties and take appropriate measures).
- Take into account the experience and culture of developers and the organization.
- Choose the appropriate methodology.
- Reduce the risks of shifting from a traditional to an agile approach or from a traditional to a broad-based agile approach
- Educate developers on the importance of a critical system and the consequences of its

failure.

- Determine appropriate testing techniques (organizing workshops to predict future problems and common errors, making questionnaires for all those involved in developing the system about expected problems and their solutions without mentioning names in the questionnaire to be free from any pressure that may be exercised for fear of testing costs, then the most important of these questionnaires are discussed in the workshops). Involve developers and the project management team in testing.
- Take into account the documentation process to ensure traceability concerning ASDM methodologies.
- Take into account the application of international and local standards appropriate to the environment of the critical system.
- Hire developers with previous experience in developing such a critical system.

5. Conclusion

In this research, we reviewed many studies and research papers to find out the most prominent reasons that may lead to the failure of the development of critical systems and found that most of the causes of failure occur at the development stage. Therefore, it is necessary to use methodologies and frameworks that ensure the quality of development processes of critical systems. Despite the concerns about the transition from the use of traditional methodologies to Agile methodologies for the development of Biosystems, the business market is strongly moving towards the use of Agile methodologies to achieve quality. The study focused on addressing the most prominent ASDMs such as SAFe, LeSS, and Scrum, in addition to the traditional PMBOK methodology, and found that each methodology has advantages and disadvantages. The study also found that choosing the appropriate ASDM can contribute to reducing the failure of critical systems. However, there is no single methodology that can meet the requirements of all critical systems because each critical system has its own critical characteristics and features, which must be carefully considered. A careful study of the monetary system is essential to identify critical characteristics of the system, including identifying challenges and taking appropriate measures. It is important to consider the experience and culture of the developers and the organization. Choosing the appropriate methodology is crucial. Can the organization minimize the risks of transitioning from traditional to agile approaches, or from traditional to wide-scale agile approaches? To reduce the possibility of failure in developing critical systems in addition to taking advantage of the advantages provided by Agile software development methods, organizations must consider the possibility of using both agile and traditional approaches in a single framework that takes advantage of the advantages of both approaches to developing critical systems. Here it is worth noting that the PMBOK is considered ideal in the administrative aspect of project management, taking into account the documentation and tracking process, providing good techniques and methods for developing critical systems, and adhering to the quality standards required for critical systems development projects. Scrum is considered easy to integrate with

other methodologies and achieves the benefits and advantages of the agile and iterative approach in addition to being clear and uncomplicated. Educating developers about the importance of the critical system and the consequences of its failure is necessary. Identifying suitable testing techniques, such as organizing workshops to predict future issues and common errors, conducting surveys for all system development stakeholders to identify expected problems and solutions (while maintaining anonymity to alleviate any cost-related pressures), and discussing the survey results in workshops are important steps. Involving developers and the project management team in the testing process is recommended. Documentation should be considered to ensure traceability concerning ASDM methodologies. Adhering to appropriate international and local standards for the critical system environment is crucial. Employing experienced developers with prior experience in developing such a critical system is advisable.

References

[1] K. L. Hobbs, M. L. Mote, M. C. Abate, S. D. Coogan, and E. M. Feron, "Runtime Assurance for Safety-Critical Systems: An Introduction to Safety Filtering Approaches for Complex Control Systems," *IEEE Control Syst. Mag.*, vol. 43, no. 2, pp. 28–65, 2023.
[2] J. Gogoll, N. Zuber, S. Kacianka, T. Greger, A. Pretschner, and J. Nida-Rümelin, "Ethics in the software development process: from codes of conduct to ethical deliberation," *Philos. Technol.*, pp. 1–24, 2021.
[3] K. S. Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley, 2012.
[4] C. Larman and B. Vodde, "Large-Scale Scrum: More with LeSS." Addison-Wesley Professional, 2016.
[5] F. Almeida and E. Espinheira, "Adoption of Large-Scale Scrum Practices through the Use of Management 3.0," *Informatics*, vol. 9, no. 1, Art. no. 1, Mar. 2022, doi: 10.3390/informatics9010020.
[6] F. Almeida and E. Espinheira, "Large-scale agile frameworks: a comparative review," *J. Appl. Sci. Manag. Eng. Technol.*, vol. 2, no. 1, pp. 16–29, 2021.
[7] A. A. Sherstobitova, L. V. Glukhova, E. V. Khozova, and R. K. Krayneva, "Integration of agile methodology and PMBOK standards for educational activities at higher school," in *Smart Education and e-Learning 2020*, Springer, 2020, pp. 339–349.
[8] A. Maurya and D. Kumar, "Reliability of safety-critical systems: A state-of-the-art review," *Qual. Reliab. Eng. Int.*, vol. 36, no. 7, pp. 2547–2568, Nov. 2020, doi: 10.1002/qre.2715.
[9] H. S. Birkel, J. W. Veile, J. M. Müller, E. Hartmann, and K.-I. Voigt, "Development of a Risk Framework for Industry 4.0 in the Context of Sustainability for Established Manufacturers," *Sustainability*, vol. 11, no. 2, Art. no. 2, Jan. 2019, doi: 10.3390/su11020384.
[10] A. A. Thawaba, A. A. Ramli, and Mohd. F. Md. Fudzee, "PM-ISD Traceability Metrics Enhance Reliability Assessment for Safety-Critical Systems Development Processes: A Case Study of Oil and Gas Well Drilling Project," in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Oct. 2021, pp. 1–6. doi: 10.1109/ICECCME52200.2021.9590906.
[11] K.-L. Lu and Y.-Y. Chen, "Model-based design, analysis and assessment framework for safety-critical systems," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Jun. 2021, pp. 25–26. doi: 10.1109/DSN-S52858.2021.00023.
[12] P. Lohmüller, J. Rauscher, and B. Bauer, "Failure and Change Impact Analysis for Safety-Critical Systems," in *Business Modeling and Software Design*, B. Shishkov, Ed., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2019, pp. 47–63. doi: 10.1007/978-3-030-24854-3_4.
[13] A. A. Thawaba, A. A. Ramli, M. F. M. Fudzee, and J. Wadata, "Characteristics for Performance Optimization of Safety-Critical System Development (SCSD)," *J. Adv. Comput. Intell. Intell. Inform.*, vol. 24, no. 2, pp. 232–242, 2020, doi: 10.20965/jaciii.2020.p0232.

[14] E. J. R. da Silva, "Safety-Critical Applications in a Multicore Environment," Universidade do Minho (Portugal), 2021.
[15] K.-L. Lu and Y.-Y. Chen, "Safety-Oriented System Hardware Architecture Exploration in Compliance with ISO 26262," *Appl. Sci.*, vol. 12, no. 11, Art. no. 11, Jan. 2022, doi: 10.3390/app12115456.
[16] P. Kumar, L. K. Singh, and C. Kumar, "Model Based Verification of Safety-Critical Systems," in *2021 2nd International Conference for Emerging Technology (INCET)*, 2021, pp. 1–9. doi: 10.1109/INCET51464.2021.9456353.
[17] N. Papakonstantinou, J. Linnosmaa, A. Z. Bashir, T. Malm, and D. L. V. Bossuyt, "Early Combined Safety - Security Defense in Depth Assessment of Complex Systems," in *2020 Annual Reliability and Maintainability Symposium (RAMS)*, Jan. 2020, pp. 1–7. doi: 10.1109/RAMS48030.2020.9153599.
[18] R. Arora, A. G. Aggarwal, and R. Mittal, "Testing-Effort Dependent Software Reliability Assessment Integrating Change Point, Imperfect Debugging and FRF," in *Strategic System Assurance and Business Analytics*, P. K. Kapur, O. Singh, S. K. Khatri, and A. K. Verma, Eds., in Asset Analytics. , Singapore: Springer, 2020, pp. 477–489. doi: 10.1007/978-981-15-3647-2_34.
[19] M. Catelani, L. Ciani, and M. Venzi, "RBD Model-Based Approach for Reliability Assessment in Complex Systems," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2089–2097, Sep. 2019, doi: 10.1109/JSYST.2018.2840220.
[20] M. Berk, O. Schubert, H.-M. Kroll, B. Buschardt, and D. Straub, "Reliability Assessment of Safety-Critical Sensor Information: Does One Need a Reference Truth?," *IEEE Trans. Reliab.*, vol. 68, no. 4, pp. 1227–1241, Dec. 2019, doi: 10.1109/TR.2019.2923735.
[21] D. Ling, B. Liu, and S. Wang, "A component-based software reliability assessment method considering component effective behavior," in *2017 Second International Conference on Reliability Systems Engineering (ICRSE)*, Jul. 2017, pp. 1–6. doi: 10.1109/ICRSE.2017.8030748.
[22] M. Y. Shestopalov, D. I. Kaplun, A. M. Sinitca, and K. M. Zheronkin, "Software Development for Reliability Assessment and Optimization of Systems with Spare Parts," in *2019 III International Conference on Control in Technical Systems (CTS)*, Oct. 2019, pp. 249–252. doi: 10.1109/CTS48763.2019.8973241.
[23] H. Okamura and T. Dohi, "Towards comprehensive software reliability evaluation in open source software," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*, Nov. 2015, pp. 121–129. doi: 10.1109/ISSRE.2015.7381806.
[24] M. Krasich, "Modeling of SW reliability in early design with planning and measurement of its reliability growth," in *2015 Annual Reliability and Maintainability Symposium (RAMS)*, Jan. 2015, pp. 1–6. doi: 10.1109/RAMS.2015.7105066.
[25] S. Park, T. Kim, T. Lee, and S. Noh, "SIRIUS: Systematic Investigation for Reliability Improvement Upon Software," in *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, Oct. 2017, pp. 86–87. doi: 10.1109/ISSREW.2017.55.
[26] L. T. Heeager and P. A. Nielsen, "A conceptual model of agile software development in a safety-critical context: A systematic literature review," *Inf. Softw. Technol.*, vol. 103, pp.

22–39, 2018.
[27] F. Swanepoel, "The development of a hybrid Agile Software Development Methodology through the integration of Agile Software Development Methodologies with Project Management Methodologies," North-West University (South Africa), 2021.
[28] H. Maqsood, "Agile Software Development Methodologies for Safety Critical Systems," 2022.
[29] G. Islam and T. Storer, "A case study of agile software development for safety-Critical systems projects," *Reliab. Eng. Syst. Saf.*, vol. 200, p. 106954, Aug. 2020, doi: 10.1016/j.ress.2020.106954.
[30] G. F. Frederico, "Project Management for Supply Chains 4.0: A conceptual framework proposal based on PMBOK methodology," *Oper. Manag. Res.*, vol. 14, no. 3–4, pp. 434–450, 2021.
[31] A. Soltanzadeh, M. Mahdinia, A. Omidi Oskouei, E. Jafarinia, E. Zarei, and M. Sadeghi-Yarandi, "Analyzing Health, Safety, and Environmental Risks of Construction Projects Using the Fuzzy Analytic Hierarchy Process: A Field Study Based on a Project Management Body of Knowledge," *Sustainability*, vol. 14, no. 24, Art. no. 24, Jan. 2022, doi: 10.3390/su142416555.
[32] S. Buehring, "Prince2® vs PMBOK® Guide: A comparison," *Knowl. Train*, 2019.
[33] N. Takagi and J. Varajão, "Success Management and the Project Management Body of Knowledge (PMBOK): An Integrated Perspective," *Int. Res. Workshop IT Proj. Manag. 2020*, p. 11, Dec. 2020.
[34] A. R. Chaudhari and S. D. Joshi, "Study of effect of Agile software development Methodology on Software Development Process," in *2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Aug. 2021, pp. 1–4. doi: 10.1109/ICESC51422.2021.9532842.
[35] Adrielle Cristina Sassa, Isabela Alves de Almeida, Tábata Nakagomi Fernandes Pereira, and Milena Silva de Oliveira, "Scrum: A Systematic Literature Review," Institute of Integrated Engineering and Management, Federal University of Itajuba (Unifei), Brazil, Itabira, 2023.
[36] Ken Schwaber and Jeff Sutherland, "Home | Scrum Guides," *SCRUM Guid.*, p. 14, Nov. 2020.
[37] P. Adi, "Scrum Method Implementation in a Software Development Project Management," *Int. J. Adv. Comput. Sci. Appl.*, vol. 6, Sep. 2015, doi: 10.14569/IJACSA.2015.060927.
[38] A. K. M. Islam and D. Ferworn, "A Comparison between Agile and Traditional Software Development Methodologies," *Glob. J. Comput. Sci. Technol.*, pp. 7–42, Dec. 2020, doi: 10.34257/GJCSTCVOL20IS2PG7.
[39] H. Edison, X. Wang, and K. Conboy, "Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review," *IEEE Trans. Softw. Eng.*, vol. 48, no. 8, pp. 2709–2731, Aug. 2022, doi: 10.1109/TSE.2021.3069039.
[40] S. Nilsson Tengstrand, P. Tomaszewski, M. Borg, and R. Jabangwe, "Challenges of Adopting SAFe in the Banking Industry – A Study Two Years After Its Introduction," in *Agile Processes in Software Engineering and Extreme Programming*, P. Gregory, C. Lassenius, X.

Wang, and P. Kruchten, Eds., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2021, pp. 157–171. doi: 10.1007/978-3-030-78098-2_10.
[41] R. Knaster and D. Leffingwell, *SAFe 5.0 distilled: achieving business agility with the scaled agile framework*. Addison-Wesley Professional, 2020.
[42] S. Block, "How to Adapt and Implement a Large-Scale Agile Framework in Your Organization," in *Large-Scale Agile Frameworks: Agile Frameworks, Agile Infrastructure and Pragmatic Solutions for Digital Transformation*, Springer, 2023, pp. 65–168.
[43] A. Putta, M. Paasivaara, and C. Lassenius, "How Are Agile Release Trains Formed in Practice? A Case Study in a Large Financial Corporation," in *Agile Processes in Software Engineering and Extreme Programming*, P. Kruchten, S. Fraser, and F. Coallier, Eds., in Lecture Notes in Business Information Processing. Cham: Springer International Publishing, 2019, pp. 154–170. doi: 10.1007/978-3-030-19034-7_10.
[44] P. Mishra, "Quality Deployment and Use of the Scaled Agile Framework®-Managing teamwork and software quality in the banking sector," Master's Thesis, University of Tampere, 2018.
[45] Ö. Uludag, "Empirical Analysis of the Adoption of Large-Scale Agile Development Methods," A dissertation for an academic degree, Technical University Munich, Munich, 2022.
[46] C. T. Ranner, "Large-scale Agile Software Development: An Exploratory Case Study of Changes In Agile Practices and Its Effects on Inter-team Coordination," NTNU, 2020.
[47] V. Malhotra, *Single Reference Guide for Scrum Certification: Professional Scrum Master I (PSM I) and Professional Scrum Product Owner I (PSPO I) Certification*. Vishal Malhotra, 2020.
[48] B. Kischelewski and J. Richter, "Implementing large-scale agile-an analysis of challenges and success factors," 2020.