education sciences

MDPI

Article

Software Development Methodologies, HEIs, and the Digital Economy

Kawther Saeedi [1,*] and Anna Visvizi [2,3]

[1] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21551, Saudi Arabia
[2] Effat College of Business, Effat University, Jeddah 21551, Saudi Arabia; avisvizi@gmail.com
[3] School of Business & Economics, Deree College, The American College of Greece, 153-42 Athens, Greece
* Correspondence: ksaeedi@kau.edu.sa; Tel.: +966-12-695-2000

Abstract: Progressing digitalization of business, economy, and the society places higher education institutions (HEIs) in the center of the debate on how to effectively respond to challenges and opportunities that are thus triggered. Several facets of this process and corresponding challenges exist, including the complex question of how to match students' skills and competencies with the demands and expectations of the industry. From a different angle, considering the changing nature of work, HEIs are responsible for equipping future employees with skills necessary to work in virtual, distributed, culturally diverse, and frequently global, teams. In the domain of software development, i.e., the backbone of the digital world, the challenge HEIs need to face is paramount. For this reason, the way software development is taught at HEIs is crucial for the industry, for the economy, for the students, and for the HEIs. As there is a tendency in the industry to embrace the scrum method and seek employees equipped with skills necessary for the scrum methodology use, it is necessary to ensure that HEIs offer the students the opportunity to get exposed to scrum. By querying the challenges of switching to agile software development methodologies in senior capstone projects, this paper makes a case that software development and software development methodology form the thrust of a multi-stakeholder ecosystem that defines today's digital economy and society. In this context, the added value of this paper rests in the elaboration of a method enabling HEIs to move toward scrum in senior projects.

Keywords: digital economy; education; agile software development methodologies; scrum; waterfall; higher education institutions; industry-academia collaboration

Citation: Saeedi, K.; Visvizi, A. Software Development Methodologies, HEIs, and the Digital Economy. *Educ. Sci.* **2021**, *11*, 73. https://doi.org/10.3390/educsci11020073

Academic Editor: Hironori Washizaki

Received: 11 January 2021
Accepted: 10 February 2021
Published: 13 February 2021

1. Introduction: The Context and the Case Study

Progressing digitalization of business, economy, and the society places higher education institutions (HEIs) in the center of the debate on how to effectively respond to challenges and opportunities that are thus triggered [1,2]. Several facets of this process and corresponding challenges exist, including the complex question of how to match students' skills and competencies with the demands and expectations of the industry. Indeed, the three key dimensions of that question are (i) whether the graduating students will find employment, (ii) whether the industry, by employing these graduates, will retain its competitive advantage, or perhaps will be able to build it, and (iii) whether a given HEI, the alma mater of these graduates, will thus be considered competitive enough to attract the best future students and, therefore, funding. From a different angle, considering the changing nature of work, a process dramatically accelerated during the Covid-19 pandemic [3,4], HEIs are responsible for equipping future employees with skills necessary to work in virtual, distributed, culturally diverse, and frequently global, teams [5]. In the domain of software development, i.e., the backbone of the digital world, the challenge HEIs need to face is paramount. For this reason, the way software development is taught at HEIs is crucial for the industry, for the economy, for the students, and for the HEIs. In

other words, approaches to software development and software development methods form the thrust of a multi-stakeholder ecosystem that defines today's digital economy and society [6,7].

Even if the discussion on ways and approaches to teaching software development in HEIs has continued for more than a decade now [8–10], it is vital that this debate is revisited. To do so, this paper examines the opportunities and caveats of software development training offered to senior students, especially as seen from the angle of two approaches to software development, i.e., waterfall and scrum. It is argued that while the waterfall approach has some advantages, agile methods, including scrum, create the opportunity, on the one hand, to create a close-to-real world work environment for students involved in a software development project, and on the other hand, to equip students with skills required by the industry, and the labor market in general. Indeed, insights from academic research [8,11–14] and the industry [10], also as reflected in most job descriptions and skills sought, suggest that scrum has established itself as the preferred software development methodology. Certainly, in the context of HEIs, the question of which approach to software development to prefer is a complex issue in that it weighs heavily on the organization and coordination of the project delivery [11–13], on the alignment with the academic calendar, on curriculum requirements, on accreditation obligations, as well as on faculty preferences and time-availability. For these reasons, the question is not really whether to continue applying the waterfall method, but rather how to successfully transition to hybrid or agile methods, including for instance scrum [14–17]. By reference to the case of senior software development project implemented at the King Abdulaziz University (KAU), the objective of this paper is to highlight a possible pathway that HEIs might follow to do so.

Senior software development project is a required milestone for all computer science students. Students embark on the senior project during the final year of their bachelor-level studies at the university. The objective of this milestone is to apply the knowledge learned and skills acquired throughout the program to successfully engage in a software development project. This may include the development of a prototype of a software system, a mobile application, a web application, a drone application, or any other form of a software system. Students are encouraged to develop an innovative idea, to solve a real word problem or, to develop a software that will create a broadly conceived opportunity of generating a positive social impact, i.e., through identifying a market niche or through social innovation [18–20]. To this end, students should be able to adopt and integrate in their project advanced concepts and latest technological developments, such as Artificial Intelligence (AI), Internet of Things (IoT), Blockchain, cloud computing and big data. To boost the students' engagement with the project as well as to increase the students' exposure to developments on the market, students are encouraged to participate in open local, regional, and national software development competitions [21,22]

At KAU, the senior project is a two semester, 5-credit (1 + 4 credits) bearing module that is compulsory for all students graduating in computer science. The senior project is designed in a manner that, on the one hand, allows the students to draw from, to exploit and to adopt all skills and knowledge they progressively acquired during the previous semesters. On the other hand, it allows to create evidence that students acquired practical skills matching the market's/industry's expectations and demands. For this reason, as a part of the course assessment, the students are required to develop a program or a software system, to build a corresponding software, and to supply the necessary documentation of the work done, especially as regards the software engineering process. Students are evaluated based on the presentation of the project outcomes to the faculty members as well as on a detailed assessment of the project documentation.

Until recently, the senior software development projects at KAU would follow the traditional, plan-driven sequential process, most closely associated with the waterfall method. It consisted of five phases, including analysis, design, coding, report writing and presentation. These phases were mirrored in the assessment and the evaluation rubrics. Given the developments in the global economy, but also considering the objectives

outlined in the Vision 2030 [23,24] a consensus begun to consolidate that a switch to agile methods might be of benefit to the students, the university, and the economy at large. This nascent understanding emerging among the faculty and the administration at KAU mirrors tendencies and trends present elsewhere in the world [25–27]. In this sense, the findings elaborated in this paper are generalizable and might be of use to educators and administrators around the world. The reminder of this paper is structured as follows. The second section offers a brief insight into the traditional and agile software development methods to highlight their added value and potential. In what follows, the agile software development methods are mapped. Against this backdrop, the specific features of scrum are elaborated. The next section returns to the case-study and examines the specific stages of the senior project implementation to highlight the nodal points where challenges and opportunities are generated. A blueprint for the transition to agile software development methods is elaborated. Discussion and conclusions follow.

2. Delineating Traditional and Agile Software Development Methods

Software development life cycle (SDLC) is one of the key issues in software engineering. It is because the structure of activities required to develop a software system weighs on the quality, usefulness and usability of the software developed [28,29]. Several approaches to software development exist. A generic SDLC consists of five phases including: requirements analysis, software design, software implementation, and software testing. In brief, the requirements analysis phase is the process of capturing user requirements and producing software specifications. However, variations exist [28]. During the software design phase, abstract software models are developed through illustration of different perspectives using modeling languages such as the Unified Modelling Language (UML) [30,31]. The software implementation phase is the process of building the executable software based on models built in the earlier design phase. The software testing phase is meant to ensure that the developed software meets prior specifications and is free from errors/bugs.

Two key approaches to the workflow in SDLCs exist, i.e., the linear, plan-driven, or traditional one, frequently referred to as waterfall, and the non-linear, an incremental one, frequently referred to as agile. It would be a simplification to argue that one approach is better or worse. The practice of software development projects suggests that elements of both plan-driven and incremental methods are applied throughout the SDLC. However, evidence suggests that more than 70 percent of organizations use agile software development method [32,33] in which the delivery of functional sub-components of the entire project is placed in the forefront.

The waterfall method is a plan-driven software development methodology based on advanced planning of all project phases, which are subsequently implemented. Moving from one phase to another requires a verification and formal approval from the software customer. This approach to software development is useful to coordinate the work of a big project among different teams. The drawback of waterfall is that it is difficult to accommodate necessary changes, possible and necessary corrections and/or adjustments as the project and its phases unfold. In a way, a certain degree of path-dependence is inherent in the waterfall method. Indeed, the customer's feedback is received at the end of the whole process, which adds to the challenge of customizing the software developed along the way. In this view, the waterfall method is a method suitable in all those (rare) cases when all functional and nonfunctional requirements are clear, well-understood, and predictable.

The agile, or incremental method (manifesto), is based on an incremental delivery of software functions, i.e., the software is built step-by-step, whereby—at each of the subsequent stages of the project implementation—the outcome of the process is assessed against the feedback received also from the customer. In other words, unlike in waterfall, feedback is fed into the process instantly and incrementally rather than only at the very end of the process [28]. As a result, there is a flexibility to go back and forth to address likely errors, inaccuracies etc. in the software under development [34]. The agile method requires

advanced teamwork and related skills. The workflow is divided into so called sprints, i.e., time-boxed units of work during which a specific sub-component of the project is to be delivered. Deliverables are prioritized according to their business value, i.e., essentially, according to the customer's specifications and needs. If all planned work for a given sprint cannot be completed, work is reprioritized, and the information is used for future sprint planning.

Agile manifesto was first proposed as an alternative to the existing software development methods, such as waterfall, spiral or V-shaped model [28,34]. The descriptor "manifesto" is to be associated with a set of values and principles upon which the agile method is built. The four values include: (i) value the individuals and the interactions over processes and tools, (ii) value the working software, (iii) value the customer collaboration, and (iv) and value the process of responding to change [34]. The four principles include: (i) work to achieve better quality software, (ii) fast delivery on the market, (iii) responsiveness to the requirements of change and (iv) less risk of failure [35]. These principles promote active customer involvement from early stages of the project as well as the accommodation of changing requirements so long it is aligned with the customer's competitive advantage [26,36,37].

A project starts with addressing customer's highest priority requirements in a working software, it follows the customer evaluation of the developed requirements, and accommodates the corresponding feedback of the customer. In this approach, the customer benefits from the software already at the early stage of the project, which is essential in view of preempting likely errors/bugs and misfit as regards the customer's requirements. The incremental approach and active participation of the team and the customer foster trust between them. This is important for project success [38].

The agile manifesto values and principles are generic and can be adopted by many methods or frameworks such as scrum [39], Extreme Programming (XP) [40], Crystal [41], Feature Driven Development [41], Kanban [42], and other methods. These are reviewed in the following section. Then, the spotlight is directed specifically at scrum. In the following step, the experiences gained during the application of the agile framework are elaborated.

3. Mapping the Agile Software Development Methods

This section reviews the most common agile software development methods to identify their specificity and value added, especially in view of challenges associated with the digital economy and industry-academia collaboration. The following frameworks are reviewed: Extreme Programming (XP), crystal methodology, Feature Driven Development (FDD), Kanban method, and Scrum. After the brief highlight on most popular agile methods for software development, Table 1 (beneath) offers a summary of each of these frameworks, including: team size, roles, iteration (sprint) length, release, release status, large project adaptation, value driven development and Iteration plaining.

Table 1. Overview of the agile software development methodologies.

	Scrum	FDD	XP	Kanban	Crystal
Team Size	**3–9 Members**	**Not Specified**	**Up to 12**	**Not Specified**	**Different Team Size**
Roles	Fixed roles include: - Scrum master - product owner - Development team	Various may include: - Class owner - Feature team - Chief member - Chief programmer - Chief Architect	Mandatory: - Customer - Programmer - Coach Optional: - Tester - Tracker - Manager	- Project manager - Team member	Optional roles: - Executive - Lead Designer - Tester - Sponsor - Team member - Coordinator - Business expert
Iteration Length	Fixed and maximum 1 month	Features based Variable 2–10 days	Variable 1 to 2 weeks	Continuous flow	From a week to 4 months

Table 1. *Cont.*

	Scrum	FDD	XP	Kanban	Crystal
Team Size	**3–9 Members**	**Not Specified**	**Up to 12**	**Not Specified**	**Different Team Size**
Release	End of each sprint	Feature build	Continuous Integration	Continues delivery	Release plan
Release status	Done statues ready for integration	Regular build	Working function	Done—every iteration	Integrate all at the end
Large project adaptation	Scaling scrum to manage different team	Cascading the project features into smaller sets	not applicable	Applying same method	Crystal clear, yellow, orange, red and maroon
Value driven development	Product backlog priority setting by product owner	Cascade features list	Story card	Stories on Kanban board priority setting by customer	Episode which reflects the iteration size
Iteration plaining	Per sprint	Per feature	Release planning Iteration paling	For each story	Every iteration
Collaboration	- Scrum events - Cross functional team - Self-organizing team	- Relay on documentation - Rare meeting	- Pair Programming - On-site customer - Coding standard	- Optional Kanban meeting - Customer priority	- Requires documentation - Frequent communication

Extreme Programming (XP) was first introduced in 1996 [40]. In this approach emphasis is placed on customer satisfaction, which is ensured through continuous feedback. This method empowers the team engaged in the development of the software to respond to changes in software requirements, as defined by the customer [43]. It involves collaboration between a small development team and the customer. The development team consist of two to ten members working on subcomponents of the software such as a problem to solve or a new functional requirement. The team members organize themselves as efficiently as possible around a problem, or a requirement, that is to be addressed. XP consist of simple rules guiding the implementation stages. These consist of planning, managing, designing, coding, and testing [40]. The designing process of XP is simplified and uses a system metaphor, spike solution to reduce the risk of uncertainty and code refactoring [43]. The coding is produced in pair programmed fashion, following agreed standards, unit tested, version controlled and continuously integrated. The testing is conducted for each code unit separate component and integrated components to ensure fixing all bugs. The XP is suitable for high-risk projects with short duration, dynamic requirements, and implemented by a small team [40,43].

Crystal methodology is a lightweight methodology i.e., minimum documentation, management and reporting. This gives the method a generic and flexible form of adaptation. The adaptation suggests different crystal methodologies depending on the project environment and team-size, including such methodologies as clear crystal, yellow shining, orange crystal and red crystal [41]. Each of these methodologies requires different practices, processes, and policies [41]. In this method, the focus is directed at human interactions where the process adaptation is based on what is suitable for the team. Software development iterations are set according to the task chosen by the team as high priority. The development process is flexible and adjusted to the development team. The crystal methodology consists of frequent process cycles of various length and order including project cycle, delivery cycle, iteration cycle and integration cycle [41]. The project cycle comprises charting project initial plan and its development. The delivery cycle comprises the release plan and presentation to real users. The iteration cycle includes iteration planning, daily activities, and reflection for improvement. Lastly, the integration cycle is continuously integrated up until the release of the software.

Feature Driven Development (FDD) as the name indicates, is a software development method in which the stress is placed of the features of the software being developed. FDD

builds around software engineering best practice. Therefore, it is suitable for complex and large-scale project [37,41]. In this framework a given software development project is split into several cascading features small enough for each feature to be executable within a timeframe of two to ten days. The FDD process flow starts with building an overall model and identifying project features set then decompose them into group of features. Each of which is realized through the iteration of six phases starting by design, design inspection, coding, unit testing, integration, code inspection and finally releasing the main build. This process is repetitive for every features of the project; parallel iteration conducted through different teams. Another important aspect of FDD is clear roles and responsibilities; there are six main roles in FDD, which includes a project manager, chief architect, Development Manager, Chief Programmer, class owner, domain expert and other supporting roles [41].

The Kanban method offers a visual workflow management framework of SDLC [42]. Kanban is based on three main practices, including: progress visualization, limiting work in progress, and workflow enhancement [42]. Progress visualization, including the work to be done, work in progress and finished work, is depicted on a physical or an electronic board. The practice of limiting work in progress essentially means that before new stage of the process begins, all pending tasks are accomplished. The practice of workflow enhancement is meant to maximize the value of delivery to the customer, to ensure smooth process and to minimize the process' cost and time. The Kanban method does not specify the roles other than the conventional project manager and the development team.

Scrum is the most popular agile framework used in the industry. Notably, it is the only agile framework included in the skills category of the most popular job seeking site of a global outreach [44]. This popularity is a function of this framework's capacity to boost the efficiency of the software development process, including the speed of delivery and the quality of the software itself. Another aspect that made this framework very popular is that it lends itself to be adopted as a way of workflow management in other business domains, i.e., other than software development [45]. That is, its use is not limited to software project management [33]. The scrum is a simple to understand and lightweight framework used to develop complex products [scrumguides.org]. The scrum is an empirical process, which is based on three pillars, such as: transparency, inspection, and adaptation [39]. As other agile methodology, scrum delivers the project in iteration of fixed intervals called sprints. Thus, to maximize the opportunity, during each sprint, the development team needs to empirically visit the three pillars. The framework consists of three scrum roles, five scrum events, three scrum artifacts and the rules that bind them together. The following section offers a detailed insight into the scrum framework and it its value for the software development process.

4. A Detailed Insight into the Scrum Method and Its Potential

The three roles specific to the composition of a scrum team are scrum master, product owner and the development team. The scrum team is cross-functional and is equipped with skills required to implement the project. The team is also self-organized, in that the ways in which the team operates is defined by the team. The product owner is responsible for conveying to the team the customer's requirements and managing the work timeline. This is done by reference to priorities assigned to each stages of the project delivery. The lists of requirements and priorities are documented in a shred backlog. The development team consists of three to nine members. This number is said to balance the complexity of communication among team members and the ability to deliver releasable increment at the end of each iteration/sprint. The development team's main responsibility is to build the increment during the sprint. The scrum master is a "servant leader", who is responsible for ensuring that scrum guidelines are applied, that the impediments the team faces are addressed, and that communication between the team and the customer is facilitated.

The five scrum events are time boxed, i.e., have maximum duration and regular at the same time and location [39]. The sprint event is the container of the other four events, where development team expected to deliver releasable increment at end of it. The sprint is

time boxed with one month or less to limit the risk. This period is fixed thought the project and new sprint start straight after the previous one. The scrum team, product owner and scrum master get together to plan the coming sprint in the sprint-planning event. They specify what can be delivered and how it will be built in the next sprint in the Sprint Backlog. The sprint planning is eight hours time box for one-month sprint, shorter sprints have shorter planning hours. Daily scrum is a daily time-boxed event that lasts 15 min and is designed to assess the progress of the development team on a daily basis. Each member of the development team answers three questions: "What did I do yesterday?", "What will I do today?", "What difficulties I am facing today?" The answers to these questions create a good background for the team to consolidate collaboration and to assess the team's performance. The sprint review held by the end of the sprint is designed to inspect the work done. The sprint review is a four-hour time box for a one-month sprint. The sprint retrospective is an inspection meeting during which the team may reflect on the team's overall performance, including people, relationships, processes, and tools. A thorough and critical evaluation of these items enables the team to draw conclusions as to how to conceptualize and strive for improvement. The retrospective event is a three h. time box for one-month sprint.

The scrum's artifacts consist of product backlog, sprint backlog, and an increment [39]. The product backlog is a single repository of the work that needs to be done. This includes new functions, improvements, big fixes, and other tasks that need to be addressed by the development team. The product owner is the main person responsible to ensure that the backlog items are clear, prioritized, and complete. The development team can add new items or refine the items already available in the backlog. This single repository of the work related to the project maximizes transparency as well as the informal progress tracking of the project. Sprint Backlog includes items from the product backlog created in the sprint planning so that the development team can tackle them during the next sprint. The items in the sprint backlog form a sprint goal, which is mirrors the objective and the scope of work for the development team. The increment is a work or product ready for release delivered by the end of each sprint. It is possible to develop multiple increments during a sprint. The increment must comply with the definition of "Done", i.e., the agreed stander for releasable product. The increment is internally inspected during the sprint review. The increments are gradually integrated in the project framework to step-by-step achieve the product goal. The release and the integration schedule are conducted through coordination with the product owner.

5. The Research Model, the Examination, and the Results

5.1. The Research Model

The data included in this analysis was collected over the period 2019–2020 and consisted of a structured evaluation of the senior project implementation documentation. Specifically, for the purpose of this examination, this research process was divided into four stages (Figure 1). (1) Examination of the specificity of the senior project coordination process. (2) Process-focused analysis of the senior projects' documentation; (3) Method-focused analysis of the projects' documentation; (4) development of a blueprint for moving away from waterfall to agile methodology. These stages are elaborated beneath.

Stage 1: The objective of this stage of research was to examine the senior project coordination process, i.e., how the senior projects are conduced and evaluated. Naturally, the coordination process mirrors the software development method. In this case, the coordination process was constructed in a manner mirroring the waterfall method. Accordingly, for the purpose of the evaluation the following features of the senior projects were assessed: (i) the coordination process, (ii) the composition of the team, (iii) the evaluation method, and, finally, (iv) the deliverables.

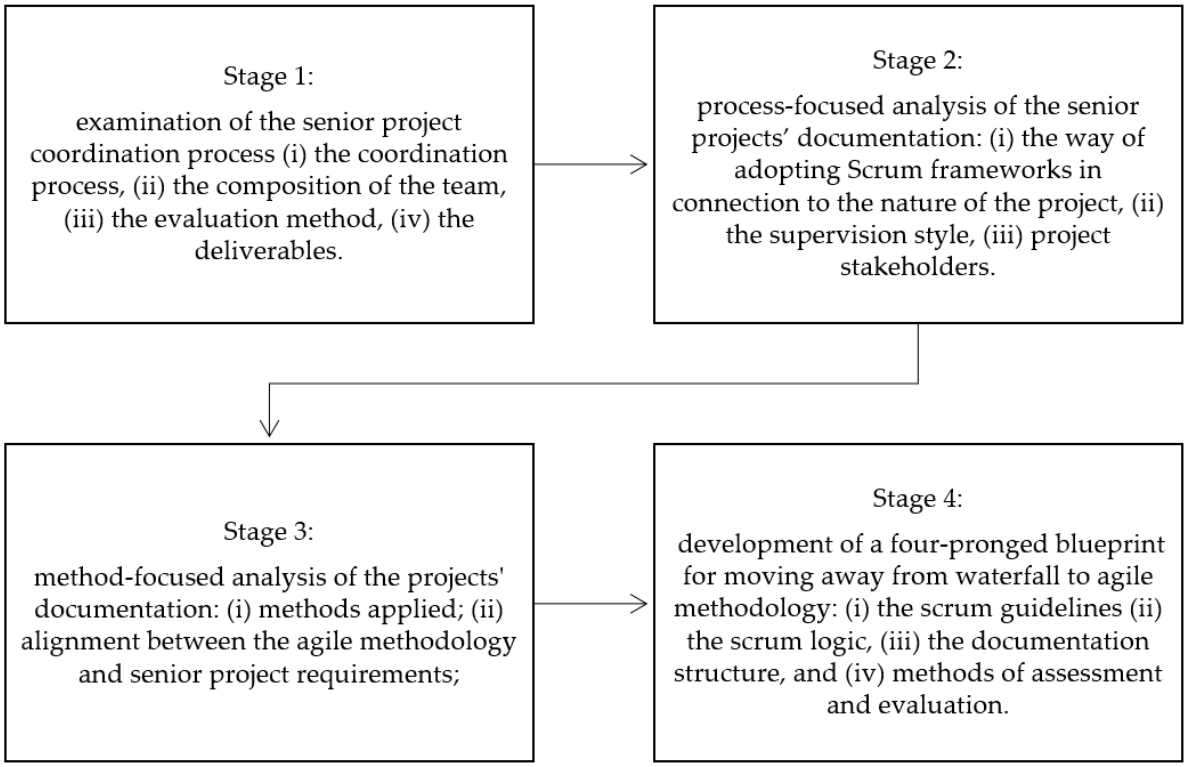

Figure 1. The research model. Source: The authors.

Stage 2: The objective of this stage of research was to conduct a process-focused analysis of the senior projects' documentation. To this end, thirty-four (34) sets of senior projects' documentation were evaluated. A three-pronged crosscheck examination of the projects was performed to identify how Scrum framework was adopted. This entailed identifying: (i) the way of adopting Scrum frameworks in connection to the nature of the project, (ii) the supervision style and (iii) project stakeholders.

Stage 3: The objective of this stage of research was to conduct a method-focused analysis of the projects' documentation, to (i) identify the methods applied; and (ii) to understand and pinpoint the degree of the possible alignment between the agile methodology and senior project requirements. In other words, the objective was to determine whether an alignment between the agile methodology and senior project requirements is feasible.

Stage 4: Based on the findings arrived at Stages 1–3, a blueprint for switching from waterfall to scrum methodology was developed. This included: (i) the road map of the process, (ii) the deliverables, (iii) the documentation structure, and (iv) the methods of assessment and evaluation blueprint. Notably, the blueprint elaborated at this stage refers to the components of the methodology proposed in this paper. As such this blueprint is generic and can be adopted in diverse educational settings.

5.2. Applying the Research Model to the Case-Study

The decision to leave waterfall behind and adopt scrum as the key senior project implementation method requires a broad horizontal consensus among the faculty members. Therefore, changes need to be implemented incrementally, and voices of concern need to be attended to. Even if this approach to scrum adoption in senior projects may seem time-consuming, in the end, it is the only way to promote accepted by all, and therefore sustainable, solutions [46,47]. As mentioned, the adoption of scrum as the key senior project software development method weighs heavily on the organization and coordination of the project delivery, on the alignment with the academic calendar, on curriculum requirements, on accreditation obligations, as well as on faculty preferences and time-availability. These

points and observations form the strategic background against which the scrum adoption blueprint is developed and elaborated in this paper. The following sections attest to that.

5.2.1. Stage 1: Examination of the Senior Project Coordination Process

Coordination

The senior project coordination process consists of several phases, including the overall oversight by the faculty, the induction of students into the project requirements, supervision and delivery of the project outcomes, and several evaluation stages. The senior project is a group project. The group may consist of two to four members. The project is implemented over the timespan of two consecutive semesters, with each semester lasting 17 weeks. Figure 2 (beneath) offers a detailed insight into the workflow related to the senior project implementation.

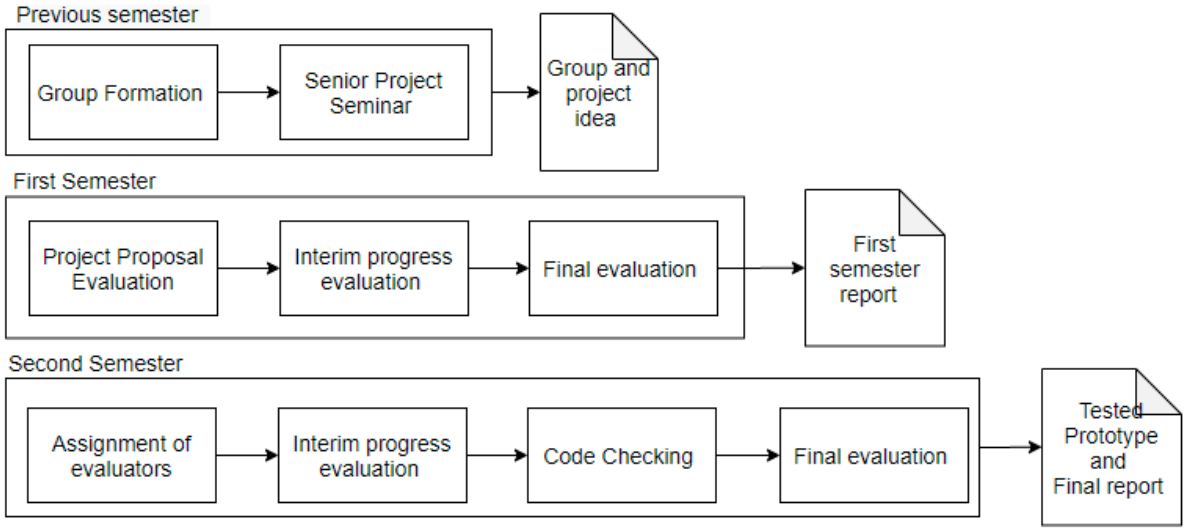

Figure 2. The senior project coordination process. Source: The authors.

The senior project is coordinated by two groups of faculty members, i.e., the senior project committee, and by the supervisors. The senior project committee is responsible for inducting the students into the overall requirements of the senior project, including the grading criteria and evaluation rubrics; for facilitating group formation and supervisors' selection; for guiding the students through the evaluation milestones; for selecting the evaluators and overseeing the evaluation process; as well as for providing feedback to students.

At this stage of the project implementation, a senior project seminar is held. Its objective is to induct students into diverse software development methods. These may have been discussed in previous semesters, nevertheless it is important that students' knowledge and understanding of the process and methods is refreshed, consolidated, and brought up to a certain benchmark. This is important if work in groups is to be feasible and efficient.

To promote a seamless move toward the adoption of agile methods, over the period 2019–2020, a few incremental changes were introduced in the delivery of the said senior project at KAU. In this context, during the senior project seminar, the students and the faculty were exposed also to agile software development methods. Here the explicit aim was to enhance students' and faculty's awareness of the agile software development methodology. In addition, the implicit rationale behind the inclusion agile methodology in the senior seminar was to acquire an insight into the faculty's views on the possibility and feasibility of moving away from waterfall and adopting agile methods, especially scrum. This included the faculty's/supervisors' opinion on how to best to adapt, to evaluate, and to manage senior projects when employing the agile software development methodology.

The Evaluation Process (Methods of Evaluation and Assessment)

An important part of the senior project implementation is the evaluation process. To ensure fair and constructive evaluation, each project is evaluated by four evaluators, including the supervisor, two faculty members and the senior project coordinator. Project evaluation is an important component of the teaching and learning process in that, ideally, it structures the teaching and learning process by highlighting the goals and objectives the learners and the educators are expected to attain [48–52]. Well-designed evaluation rubrics are an important tool that facilitates students in organizing their work and improving their efficiency. From a different angle, the said rubrics allow the instructors direct students at the attainment of very specific learning outcomes, which are congruent with the overall orientation and objectives of a given major. In the context of the senior project examined in this paper, the evaluation process consists of 5 stages because students' work is evaluated in two times in the first semester (1st interim progress evaluation and 2nd Final evaluation) and three times during the second semester (3rd interim progress evaluation, 4th code checking and 5th final evaluation activity) The goals and objectives of each of the interim evaluation activities are outlined in Figure 3 (beneath):

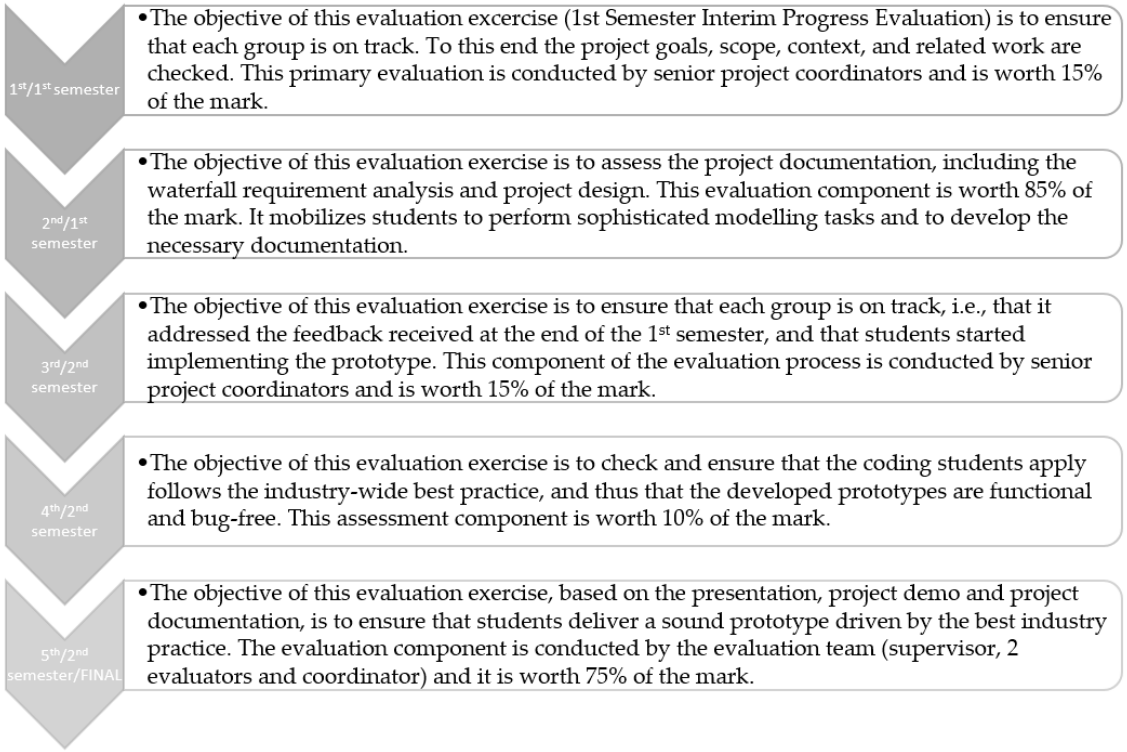

Figure 3. Senior project evaluation stages. Source: The Authors.

The final output of the project implementation is assessed through an 8-item evaluation rubric based on a 1 to 5 scores. The 8 items include:

- Problem definition and identification of aims, i.e., identification of the project's rationale and relevance.
- Literature Review, i.e., a comprehensive and critical review of relevant academic and empirical sources linked to the project.
- Methodology, i.e., identification of a relevant methodology and a thorough justification of its selection.
- Requirements and analysis, i.e., identification and analysis of all requirements.
- Initial solution design, i.e., a realistic and appropriate system design solution.

- Originality and creativity, i.e., the extent to which the work output depicts the students' ability to generate new, frequently, out-of-the-box ideas, and relating to that problem solving skills.
- Report style and format, i.e., the extent to which the report documenting the implementation of the project mirrors the established academic standards of writing and presentation.
- Presentation, i.e., the quality of the oral presentation of the project outcomes, including the ability to respond to questions posed by the project evaluators.

A few important points to consider: Until now, the senior project methods of assessment and evaluation follow the sequence of phases specific to the waterfall methodology. This is because waterfall: (a) it is easily adjustable to the flow of the academic year and corresponding semesters; (b) it is predictable, i.e., by completing a semester, students complete also a stage in the project implementation; (c) it is conducted in a safe setting of the university, essentially in disregard of developments in the outside world, including the industry; (d) the "requirements" are limited to those "imagined" by students and their supervisors, rather than specified by a client/customer; (e) there is no need to go back and forth to respond to client's/customer's requests/requirements; (f) the documentation prepared at the end of the first semester, continues to be progressively built throughout the 2nd semester. This means that students and their supervisors get a safe handle of the deliverables, regardless of the possibility that the documentation does not mirror the changing context, in which the software is developed.

5.2.2. Stage 2: Process-Focused Analysis of the Senior Projects' Documentation

The objective of this stage is to identify (i) the way of adopting scrum framework in connection to the nature of the project, (ii) the supervision style, (iii) project stakeholders. Accordingly, the data collected over the period 2019–2020 from a dataset consisting of thirty-four (34) senior projects' documentations, revealed the following

(i) The way of adopting scrum in senior projects

Among the thirty-four projects, twenty-three (23) projects, i.e., 68% of all projects implemented, followed the scrum methodology to develop a variety of applications including a web system, a mobile application, and an IoT devise. Most of those projects utilized advanced computation topics, or state of the art technology, such as blockchain, drones, IoT, virtual reality (VR), augmented reality (AR), Artificial Intelligence (AI) and Location-based services (LBS). Please, refer to Table 2 (below) for a summary. Please, note, that for a variety of reasons, for instance the project type, but also convenience, the remaining project teams adopted other methodologies, including also waterfall. At the end of the year, the students were able to demonstrate a tested prototype, or a minimum viable product (MVP), for innovative solutions in various domains such as healthcare, transportation, special needs, education, and other. Table 3 depicts the details.

Important points to consider: Even if all students were given the opportunity to attend the workshop on scrum framework development and utilization, as mentioned above, 68% of groups decided to follow this approach in their senior software development project over the past two years (2019–2020). The remaining 32% of the projects followed different methodologies such as waterfall, test driven development, extreme development, or for instance, data science project cycle. The choice of methodology was related to the project type and the preference of the project team. For example, expectedly, a research project based on data analysis followed a data science project cycle. However, reasons of convenience may have also played a role in a team's selection of the project methodology, such as waterfall. Clearly, despite waterfall's limitations, it offers straightforward phases easy to follow. Still, that over the past two years, 68% of project teams decided to follow the scrum framework suggests a high perceived convenience and effectiveness of scrum vis-à-vis its applicability in senior projects. It also highlights the salience of supporting methodology for senior projects.

Table 2. The kind and frequency of specific technology used in senior projects.

Technology	Blockchain	IoT	Drone	AI	VR	AR	Gaming	LBS	Total
# of projects	2	5	1	5	1	2	5	13	34

Table 3. Scrum framework adoption in senior projects.

Scrum Components	Scrum Framework Guidelines	Scrum Adaptation in Senior Project
Scrum master	Ensuring that scrum guidelines are in acted, remove impediment facing the team and facilitate communication outside the team.	43% A student appointed as scrum master
		13% The supervisor appointed as scrum master
		39% Role is not specified
Product owner	convey customer requirements to the team	13% Customer appointed as product owner
		22% Supervisor appointed as product owner
		13% Student appointed as product owner
		8% A student & supervisor appointed as product owner
		43% Role is not specified
Development team	3 to 9 members build the increment during the sprint.	100% Applied
Product backlog	A single repository of the work	100% A single repository of the work
	Priorities delivering value to customer.	56% Priorities by value to the customer 13% Priorities by story size 31% No priority is defined
	Product owner responsibility	100% Student responsibility
Sprint backlog	Output of the plaining evet, List of stories with common objective team going to work on next sprint.	8% Identify task in sprint backlog 92% Implicitly define the work of next sprint in the weekly meeting with the supervisor
Increment	The work produced by the end of each sprints	100% Weekly work of the students including project and evaluation component
Definition of "Done"	Agreed delivery status of an increment	4% Explicitly defined done property 91% Implicitly through unit testing
Sprint	Sprint time-box < month. Fixed thought the project	17% Set the sprint time box to a week 82% variable sprint period to fit the story
Sprint Planning	Plan coming sprint, what to work on and how to do it Time-box < 8 h	Plan the work of next sprint in the weekly meeting with the supervisor
		No time box specified
Daily scrum	A daily 15-min meeting held every day same place and same time to inspect the progress.	Informal exchange of the progress among the team trough phone, WhatsApp, email or in person No time box, No fix time
Sprint review	Heled by the end of the sprint to inspect the work produced. Time-box < 4 h	Review takes place in different forms Project Evaluation Code check Meeting with supervisor No time box, not every sprint and No fix time
sprint retrospective	Inspection meeting to the team themselves Time-box < 3 h	Not conducted

Source: The authors.

The specificity of scrum allows that the framework is adopted in several ways, as long as the evaluation and assessment criteria are adhered to. Interestingly, even if all groups received the same training not all scrum framework rules were followed, and substantial variation was observed as regards the specific aspects of scrum that the teams adopted.

Table 3 depicts how scrum framework is adopted in senior project. This variability is inevitable for several reasons. For instance, the projects are developed in a lab-like conditions of the university, which do not match the real-world requirements. Specifically, senior projects rarely have a known customer, therefore the priority is given to MVP development. Moreover, senior project deliverables are a prototype and a detailed documentation of software engineering artifacts. However, again, there is no customer whose requirements may be changing and need to be adhered to. Finally, students are working on the project part-time since they attend other course, while most importantly acquire skills necessary for the senior project completion simultaneously with the project development and implementation. This suggests that three key challenges exist, and the faculty and administration should be aware of them when seeking to adopt scrum horizontally. First, wrong adoption of some scrum rules may limit the benefits normally associated with the adoption of scrum. Second, the variability resulting from teams applying specific aspects of scrum creates a challenge as regards the evaluation and assessment process, especially the notions of fairness and comparability. Third, to re-create an industry-like, close-to-real working environment, the possibility of identifying a real customer may be necessary. This solution, organizationally not always feasible, would nevertheless foster industry-academia collaboration and, thus short- and long-term synergies for all stakeholders.

5.2.3. Stage 3: Method-Focused Analysis of the Projects' Documentation

This stage entails a detailed analysis of the projects' documentation to identify practices used during the projects' implementation and to determine whether an alignment between the agile methodology and senior project requirements is feasible. Scrum framework is flexible, so it is not wrong to adopt it in a variety of ways, as long as it suits the team and the project. Still, as Table 4 demonstrates, regardless of the flexibility inherent in scrum, several mistakes are committed repeatedly. These are common mistakes that refer to selected phases of scrum adoption, such as adoption the scrum roles, identification of events, artifacts, and rules, which conflict the very principles and norms of agile methodology [34]. They key reason explaining why these mistakes take place is that projects adopting the agile approach are evaluated against the same rubric which is designed for the waterfall approach. Consequently, students perform an upfront analysis and design to receive a better grade, even if this practice is against the principles and norms specific to the agile methodology [34].

Table 4. Mistakes in adopting scrum in senior projects.

Scrum Components	Common Mistakes
Roles	Scrum roles are not specified
	Appointing two members as product owner
Events	Sprint time-box is not clearly specified for a project due to fitting sprint duration to story sizes
	Lack of frequent and time-boxed events. The retrospective, review and daily sprint events are conducted as necessary
	Project plaining is conducted at early stage instead of planning the work for iteration
	Daily sprint is conducted as needed through different settings
Artifact	Product backlogs reflect the development stages, which means that waterfall is followed and not scrum
	Product backlog is organized by story size not by the value for customer
	The story estimation is linked to days of work not to the team effort
	Increments are not inspected by the end of the sprint
	The definition of "done" is missing
Rules	Scrum rules are not enacted at every iteration

Source: The authors.

5.2.4. Stage 4: Development of a Four-Pronged Blueprint for Moving away from Waterfall to Agile Methodology

Based on observations drawn from the process- and method-driven examination of senior projects' documentation (Stages 2 and 3) this section sets on to suggest a roadmap, a blueprint of the process of moving toward the adoption of scrum in senior projects. To this end the general guidelines that students should adhere to presented. Particular attention is paid to the notion of the logic underpinning scrum adoption and implementation. Equally important is to adapt the structure of the senior project documentation to the scrum specificity. Finally, the methods of senior project assessment and evaluation, including a generic rubric, are presented. It is argued that while the general scrum implementation guidelines are an important feature of a successful project implementation and delivery, it is the students' thorough understanding of the—derived from the agile manifesto [34]—logic underpinning scrum that determines the success. In terms of the project's timely and successful delivery, a careful alignment of the nonlinear scrum phases with the confines of the academic year and successive assessment and evaluation exercises is necessary. Above all, however, to ensure that the blueprint thus proposed is feasible, acceptable, accepted by the faculty and administration, and therefore also sustainable, feedback received from the faculty need to be taken into consideration. In this sense, while the specific items of the blueprint proposed beneath already mirror selected faculty's insights, the actual application of the blueprint requires discussion and moderation among the faculty members and the administration. Only in this way, their support and involvement in the process of switching to agile software development methodologies, and thereby responding to the multi-scalar challenge of the digital economy, can be ensured.

The General Guidelines that Students Should Adhere to Presented

The key assumption that educators need to embark on is that students may have several misconceptions regarding the specificity of scrum in general. Provided that waterfall may have been already in place, it is vital that students are familiarized with the scrum process. To this end, clear guidelines need to be developed on how to form the team and assign the team roles, what are the project artifacts and what are the events. Table 5 suggests how the generic guidelines could look like.

Table 5. Guidelines to adopt scrum framework in senior project.

Scrum Framework	Senior Project	Guidelines
Team		
Scrum master	Student	A member of the student team acts as the servant leader of the project in addition to the team member role.
Product owner	Customer	If available: Provide requirements, and Review Epics
	Supervisor	- Ensures that students develop main functionality and the required documentation. - Ensures that scrum is properly enacted
Development team	Students	Group of 2 to 4 student working in different locations

Table 5. *Cont.*

Scrum Framework	Senior Project	Guidelines
Artifact		
Product backlog	Requirement list	List of project requirements grouped into Epics of core functions and prioritized by important. Backlog evolves gradually as the project progresses. Each Epic is divided into stories and epic sub-functions can be developed and tested at every sprint by individual students.
Sprint backlog	Weekly work	It is a list of stories (the work to be conducted in throughout the week). Students should agree with their supervisor on the task that needs to be conducted.
Increment	- Tested function - Documentation chapters	The work developed during each sprint
Definition of "Done"	Working free from bugs and meeting the requirements	Students should unit test their work individuality then review each other work then perform integration testing
Event		
Sprint	One week	Students meet the supervisor on a weekly basis. Therefore, sprints should be weeklong throughout the semester.
Sprint review	- Weekly meeting with supervisor 1-h time-box - Weekly plaining among students	Review the work conducted during the sprint
sprint retrospective		Feedback given to the group
Sprint Planning		Plan the work for next sprint with supervisor. After the weekly meeting, students should immediately meet continue the planning meeting to decide on how the work going to be done and distribute duties
Daily scrum	Daily online meeting among students at fixed time	A daily status update on work progress agent the plan.

Source: The Authors.

In-Depth Understanding of the Scrum Logic

Apart from a clear delineation of the basic features of scrum, the students must be familiarized with the logic underpinning the process and the workflow specific to scrum. Figure 4 offers a useful visualization of the process. In this context, the challenge for the faculty is to make students understand the key logic behind scrum. To this end, students must be introduced to the basic norms and principles of the agile manifesto [34]. It is vital that these norms and principles are discussed with students at length and in-depth because the students' prospective projects' success will depend on the students' ability to implement and follow these norms and principles throughout the process. The process starts with two sequential activities, i.e., project initiation and pre-project phase, which set the foundation needed to build the rest of the project in an iterative fashion. The rest of the activities are conducted in each sprint in a sequence, and as the proposed guidelines suggested. The process can be adopted in different settings of senior project as it does not pre-define the project duration, number of semesters, project type or any other functionalities.

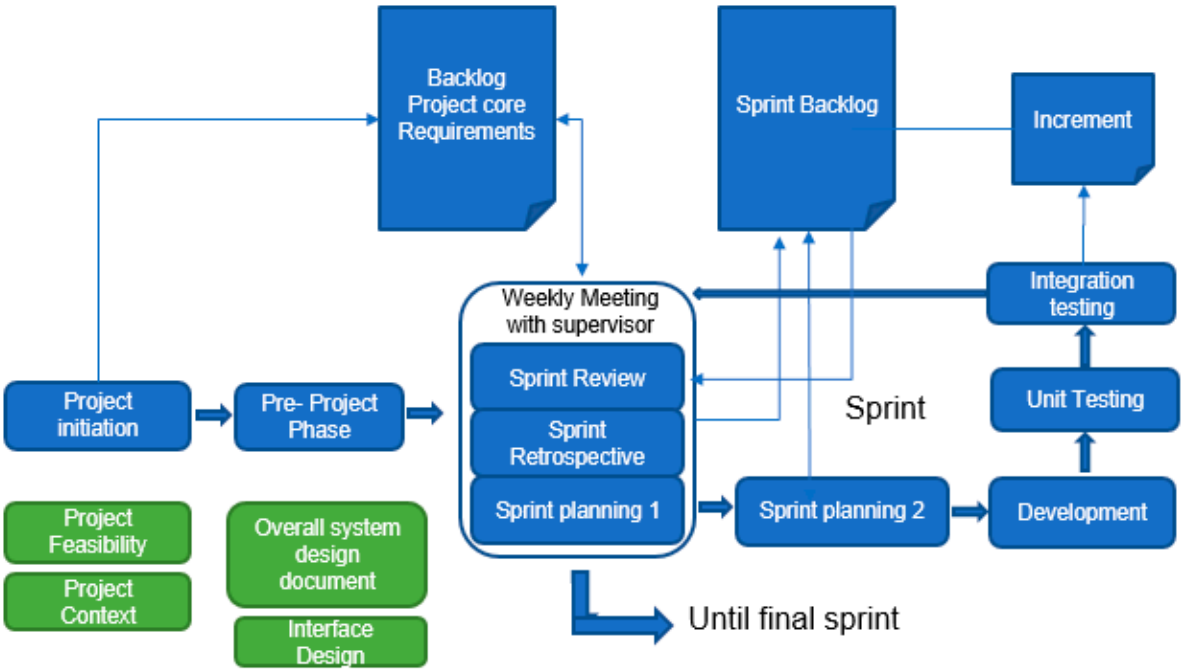

Figure 4. Visualization of the scrum process. Source: The Authors.

Methods of Evaluation and Assessment

As discussed earlier, there has been a tendency in the senior software development project implementation to rely on the waterfall method. To this end a standard linear 5-pronged modes of assessment and evaluation were adopted. Clearly, the incremental and instantaneous nature of scrum requires a more flexible way of assessment and evaluation to grant the students, i.e., the teams, the opportunity to go back and forth and to reflect on both the evolving context and the customer's specifications/requirements at each iteration/sprint. Table 6 offers a possible way of navigating the challenge of evaluating senior projects based on scrum. The rubric thus presented is very generic and, at this stage, highlights only items relevant to software development. Other items, such as students' learning outcomes and evaluation marks can be customized according to program requirements. Furthermore, the evaluation rubric does not impose either a specific number of evaluators, marks distribution or semester periods. Therefore, academic institutions wishing to use this rubric as a template have the flexibility to adjust it according to their academic requirements.

Structure of the Senior Project Documentation

One of the important requirements of HEI is documenting the project in an scientific fashion. Although documentation is not given the same important in agile manifesto, it is important to consider in academic setting as majority of the evaluation marks are based on the documentation. Therefore, this paper recommends a documentation structure in alignment with agile principles and scrum framework. Table 7 illiterate current documentation structure verses the recommended one. The current one reflect the leaner approach of software development i.e. waterfall. Whereas the recommended one reflect the iterative nature of agile manifesto. The rounded arrow implies repeating the chapter for each functional iteration such as Epic or story in scrum.

Table 6. Generic Evaluation Rubric for scrum.

	Evaluation Criteria	Unsatisfactory	Poor	Acceptable	Good	Excellent
1	Project Domain and Context					
2	Backlog of core functions ordered by priority					
3	Illustration of overall system models e.g., system architecture Context diagram and use case diagram					
4	Pre-project phase • Tools and environment • High-fidelity prototype • Database design					

		Epic development														
3	Epic Name	Epic identification			Planning			Development			Unit Testing			Integration Testing		
		1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
a	Epic One:															
b	Epic Two:															
c	Epic #:															
	Average															

1: Acceptable, 2: Good, 3: Excellent.

Table 7. Recommended documentation structure.

Current Documentation Structure	Recommended Documentation Structure
Chapter 1: Introduction chapter Chapter 2: Domain context and related work • Understanding of context • Study of similar projects, solutions • Project feasibility Chapter 2: Analysis and specification Chapter 3: Design Chapter 4: Implementation and Testing Chapter 7: conclusion and future work	Chapter 1: Introduction Chapter Chapter 2: Domain context and requirements • Understanding of context • Study of similar projects and/or solutions • Project feasibility • Backlog of key functional requirements • Context diagram • A use case diagram • System architecture Chapter 3: Project Initiation • High fidelity prototype • User interface design • ER diagram or class diagram Chapter 4, 5, 6 . . . etc. The development of Epics • Design • Implementation • Unit Testing • Integration testing (if needed) Final Chapter • System testing • Usability testing • Result and discussion • Challenges • Future work

6. Discussion and Concluding Remarks

The scrum software development methodology is the most popular agile software development framework widely adopted across the industry. However, the successful adoption of scrum for the purpose of senior software development project delivery frequently is not free from challenges, and thus requires, on the one hand, a substantial investment in institutional consensus building, and on the other hand, the development

of a blueprint facilitating the switch to scrum and its correct implementation across the board. This paper proposes a method of adopting the scrum framework in the academic context. It is important to have a method and a methodology tailor-cut for an academic context to align them with academic requirements such as evaluation, supervision engagement, working hours of students, documentation, and learning outcomes. To develop this novel methodology, senior project implementation documentation was queried. The methodology consists of four generic components, including the guidelines, the process, the evaluation rubric, and the documentation structure. The novelty of this approach proposed in this paper derives from the fact that the needs of the academic setting and the expectations of the market/industry were brought together. As mentioned earlier in this paper, even if over the years several universities switched to scrum, the relating academic debate addresses mostly such questions as team collaboration [53,54], illustration of scrum components in group project [55–58], measuring the attainment of learning outcomes [56], scrum process for specific game development [57] and scrum as a form of intervention [58]. The existing literature could be more explicit as regards the question of how scrum-based projects are evaluated, what is the appropriate documentation structure or how to make scrum, chiefly, applicable in the context of the university. This paper, by bringing together i.e., the guidelines, the process, the documentation structure and the evaluation rubric, suggested how to make scrum usable in the context of a HEI.

By means of conclusion, software development is the backbone of today's digital economy. Therefore, the way and efficiency in which students will be taught software development at HEIs will have a direct bearing on (i) the graduating students' ability to find employment [10]; (ii) on the industry, employing these graduates, capacity to retain and possibly enhance its competitive advantage; and (iii) on perceived attractiveness and competitiveness of a given HEI, the alma mater of these graduates. The prospect of adopting agile software development methodologies in senior projects is a multi-stakeholder challenge. However, as it was also argued in this paper, there is more to that. Students working on the premise of agile methodologies, especially scrum, acquire not only skills pertinent to software development, but also soft skills necessary to work in today's increasingly virtual, distributed, culturally diverse, and frequently global, teams [5]. Indeed, research suggests that students that were exposed to scrum become familiar with team dynamics, communicate effectively with IT professionals, and gain a better appreciation of the challenges involved in crafting larger and realistic software applications [59–62]. The process of applying scrum as the key methodology for senior project development at KAU suggests that students acquired and/or expanded their skills in areas such as the ability to work in a dynamic team, delivering on time and in line with requirements, collaboration with the customer. The points and observations thus outlined derive directly from the examination of the projects' documentation, from the observations of the supervisors, and the evaluators, acquired throughout the project implementation and presentation of the project results. More research is needed to offer quantitative evidence to support these points. This shall be done in our future research.

Author Contributions: Conceptualization, K.S. and A.V.; Investigation, K.S. and A.V.; Methodology, K.S. and A.V.; Writing—original draft, K.S. and A.V.; Writing—review & editing, K.S. and A.V. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Visvizi, A.; Lytras, M.D.; Sarirete, A. *Management and Administration of Higher Education Institutions in Times of Change*; Visvizi, A., Lytras, M.D., Sarirete, A., Eds.; Emerald Publishing: Bingley, UK, 2019; ISBN 9781789736281.
2. Visvizi, A.; Daniela, L.; Chen, C.h.W. Beyond the ICT- and sustainability hypes: A case for quality education. *Comput. Hum. Behav.* **2020**, *107*. [CrossRef]
3. Visvizi, A.; Lytras, M. Editorial. *Transform. Gov. People Process Policy* **2020**, *14*, 125–131. [CrossRef]
4. Sokołowski, M.M. Regulation in the COVID-19 pandemic and post-pandemic times: Day-watchman tackling the novel coronavirus. *Transform. Gov. People Process Policy* **2020**. [CrossRef]
5. Daniela, L.; Visvizi, A.; Lytras, M.D. How to Predict the Unpredictable: Technology-Enhanced Learning and Learning Innovations in Higher Education. In *The Future of Innovation and Technology in Education: Policies and Practices for Teaching and Learning Excellence*; Visvizi, A., Lytras, M.D., Daniela, L., Eds.; Emerald Publishing: Bingley, UK, 2018; pp. 11–26. ISBN 9781787565562. [CrossRef]
6. Carlsson, B. The Digital Economy: What is new and what is not? *Struct. Chang. Econ. Dyn.* **2004**, *15*, 245–264. [CrossRef]
7. Athique, A. Integrated commodities in the digital economy. *Media Cult. Soc.* **2020**, *42*, 554–570. [CrossRef]
8. Campbell, J.; Kurkovsky, S.; Liew, C.h.W.; Tafliovich, A. Scrum and Agile Methods in Software Engineering Courses. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16), Memphis, TN, USA, 15 March 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 319–320. [CrossRef]
9. Safder, I.; Ul Hassan, S.; Visvizi, A.; Noraset, T.; Nawaz, R.; Tuarob, S. Deep Learning-based Extraction of Algorithmic Metadata in Full-Text Scholarly Documents. *Inf. Process. Manag.* **2020**, *57*, 102269. [CrossRef]
10. Kellog Murray, J. The 20 Most-Requested Certifications by Employers in 2020, Indeed Career Guide, 20 November 2020. Available online: https://www.indeed.com/career-advice/career-development/most-popular-certifications-2020 (accessed on 5 February 2021).
11. Chen, C.; Hong, Y.; Chen, P. Effects of the Meetings-Flow Approach on Quality Teamwork in the Training of Software Capstone Projects. *IEEE Trans. Educ.* **2014**, *57*, 201–208. [CrossRef]
12. Baird, A.; Riggins, F.J. Planning and Sprinting: Use of a Hybrid Project Management Methodology within a CIS Capstone Course. *J. Inf. Syst. Educ.* **2012**, *23*, 243–258.
13. Rusu, A.; Swenson, M. An industry-academia team-teaching case study for software engineering capstone courses. In Proceedings of the 2008 38th Annual Frontiers in Education Conference, Saratoga Springs, NY, USA, 22–25 October 2008; pp. 4–23. [CrossRef]
14. Kisling, E. Transitioning from Waterfall to Agile: Shifting Student Thinking and Doing from Milestones to Sprints (2019). *SAIS 2019 Proc.* **2019**, *14*. Available online: https://aisel.aisnet.org/sais2019/14 (accessed on 5 January 2021).
15. Alshayeb, M.; Mahmood, S.; Aljasser, K. *Moving from Waterfall to Agile Process in Software Engineering Capstone Projects*; Nagamalai, D., et al., Eds.; ACSIT, ICITE, SIPM–2018; American Research Institute: Washington, DC, USA, 2018; pp. 107–114. [CrossRef]
16. Rover, D.; Ullerich, C.; Scheel, R.; Wegter, J.; Whipple, C. Advantages of agile methodologies for software and product development in a capstone design project. In Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, Madrid, Spain, 22–25 October 2014; pp. 1–9. [CrossRef]
17. Bastarrica, M.C.; Perovich, D.; Samary, M.M. What Can Students Get from a Software Engineering Capstone Course? In Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET), Buenos Aires, Argentina, 20–28 May 2017; pp. 137–145. [CrossRef]
18. Visvizi, A. Social Innovation in the EU and the Black Sea Region: Trends, Challenges and Opportunities. In *Empowering Civil Society in the Black Sea Region: A Tool for Innovative Social Change*; Korovesi, A., Ed.; International Centre for Black Sea Studies (ICBSS): Athens, Greece, 2013; pp. 77–87. Available online: http://icbss.org/media/1115_original.pdf (accessed on 5 January 2021).
19. Pel, B.; Haxeltine, A.; Avelino, F.; Dumitru, A.; Kemp, R.; Bauler, T.; Kunze, I.; Dorland, J.; Wittmayer, J.; Søgaard Jørgens, M. Towards a theory of transformative social innovation: A relational framework and 12 propositions. *Res. Policy* **2020**, *49*, 104080. [CrossRef]
20. Gasparin, M.; Green, W.; Lilley, S.; Quinn, M.; Saren, M.; Schinckus, C. Business as unusual: A business model for social innovation. *J. Bus. Res.* **2020**, *125*, 698–709. [CrossRef]
21. Makkah Award. Available online: https://makkahaward.com (accessed on 5 January 2021).
22. Dell Technologies Graduation Project Competition for Middle East, Russia, Africa and Turkey: Envision the Future Contest. Available online: https://emcenvisionthefuture.com (accessed on 5 January 2021).
23. Faudot, A. Saudi Arabia and the rentier regime trap: A critical assessment of the plan Vision 2030. *Resour. Policy* **2019**, *62*, 94–101. [CrossRef]
24. Ahssein Amran, Y.H.; Mugahed Amran, Y.H.; Alyousef, R.; Alabduljabbar, H. Renewable and sustainable energy production in Saudi Arabia according to Saudi Vision 2030; Current status and future prospects. *J. Clean. Prod.* **2020**, *247*, 119602. [CrossRef]
25. de Souza, R.T.; Zorzo, S.D.; da Silva, D.A. Evaluating capstone project through flexible and collaborative use of Scrum framework. In Proceedings of the 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, USA, 21–24 October 2015; pp. 1–7. [CrossRef]
26. Lárusdóttir, M.; Cajander, A.; Gulliksen, J. Informal feedback rather than performance measurements–user-centred evaluation in Scrum projects. *Behav. Inf. Technol.* **2014**, *33*, 1118–1135. [CrossRef]
27. Rodriguez, G.; Soria, Á.; Campo, M. Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to scrum. *Comput. Appl. Eng. Educ.* **2015**, *23*, 147–156. [CrossRef]

28. Sommerville, I. *Software Engineering*; Pearson: Boston, MA, USA, 2016.
29. Kneuper, R. Sixty Years of Software Development Life Cycle Models. *IEEE Ann. Hist. Comput.* **2017**, *39*, 41–54.
30. Queralt, A.; Teniente, E. Verification and validation of UML conceptual schemas with OCL constraints. *ACM Trans. Softw. Eng. Methodol.* **2012**, *21*, 1–41. [CrossRef]
31. Fernandes, J.M.; Duarte, F.J. A reference framework for process-oriented software development organizations. *Softw. Syst. Modeling* **2005**, *4*, 94–105. [CrossRef]
32. PMI. Project Management Institute Global. In *9th Global Project Management Survey, Success Rates Rise Transforming the High Cost of Low Performance*; PMI's Pulse of the Profession: Newtown Square, PA, USA, 2017. Available online: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf?sc_lang_temp=en) (accessed on 5 January 2021).
33. Sutherland, J. *Scrum: The Art of Doing Twice the Work in Half the Time*, 1st ed.; Crown Business: New York, NY, USA, 2014.
34. Fowler, M.; Highsmith, J. *The Agile Manifesto*; Project Management Initiation (PMI): Newtown Square, PA, USA, 2001. Available online: http://agilemanifesto.org/ (accessed on 5 January 2021).
35. Aziz Butt, S. Study of agile methodology with the cloud. *Pac. Sci. Rev. B Humanit. Soc. Sci.* **2016**, *2*, 22–28. [CrossRef]
36. Babb, J.; Hoda, R.; Nørbjerg, J. Embedding Reflection and Learning into Agile Software Development. *IEEE Softw.* **2014**, *31*, 51–57. [CrossRef]
37. Conboy, K.; Fitzgerald, B. Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion. *ACM Trans. Softw. Eng. Methodol.* **2010**, *20*. [CrossRef]
38. Gheorghe, A.M.; Gheorghe, I.D.; Iatan, I.L. Agile Software Development. *Inform. Econ.* **2020**, *24*, 90–100. [CrossRef]
39. Schwaber, K.; Sutherland, J. *The Scrum Guide the Definitive Guide to Scrum: The Rules of the Game*; Scrum.org: Burlington, MA, USA, 2020.
40. Fojtik, R. Extreme programming in development of specific software. *Procedia Comput. Sci.* **2011**, *3*, 1464–1468. [CrossRef]
41. Jones, C. *Software Methodologies: A Quantitative Guide*; Auerbach Publications: Boca Raton, FL, USA, 2017.
42. Anderson, D.J. *Kanban: Successful Evolutionary Change for Your Technology Business*; Blue Hole Press: Sequim, WA, USA, 2010.
43. Wells, D. Extremeprogramming, Extremeprogramming.org, 2009. [Online]. Available online: http://www.extremeprogramming.org/ (accessed on 20 November 2020).
44. Indeed, 10 October 2020. [Online]. Available online: www.indeed.com (accessed on 10 October 2020).
45. García Barriocanal, E.; Sicilia, M.A.; Sánchez-Alonso, S.; Cuadrado, J.J. Agile methods as problem-based learning designs: Setting and assessment. In Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'18), Salamanca, Spain, 24–26 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 339–346. [CrossRef]
46. Kurth, J.A.; Allcock, H.; Walker, V.; Olson, A.; Taub, D. Faculty Perceptions of Expertise for Inclusive Education for Students With Significant Disabilities. *Teach. Educ. Spec. Educ.* **2020**. [CrossRef]
47. Scott, J.A.; Halkias, D. Consensus processes fostering relational trust among stakeholder leaders in a middle school: A multi-case study. *Int. Leadersh. J.* **2016**, *8*, 54–82.
48. Baker-Shelley, A.; Van Zeijl-Rozema, A.; Martens, P. Pathways of organisational transformation for sustainability: A university case-study synthesis presenting competencies for systemic change & rubrics of transformation. *Int. J. Sustain. Dev. World Ecol.* **2020**, *27*, 687–708. [CrossRef]
49. Gallardo, K. Competency-based assessment and the use of performance-based evaluation rubrics in higher education: Challenges towards the next decade. *Probl. Educ. 21st Century* **2020**, *78*. [CrossRef]
50. Cockett, A.; Jackson, C. The use of assessment rubrics to enhance feedback in higher education: An integrative literature review. *Nurse Educ. Today* **2018**, *69*, 8–13. [CrossRef] [PubMed]
51. Hack, C. Analytical rubrics in HE. *Br. J. Educ. Technol.* **2015**, *46*, 924–927. [CrossRef]
52. Jonsson, A. Rubrics as a way of providing transparency in assessment. *Assess. Eval. High. Educ.* **2014**, *39*, 840–852. [CrossRef]
53. Klopp, M.; Gold-Veerkamp, C.; Abke, J.; Borgeest, K.; Reuter, R.; Jahn, S.; Mottok, J.; Sedelmaier, Y.; Lehmann, A.; Landes, D. Totally Different and yet so Alike: Three Concepts to Use Scrum in Higher Education. In Proceedings of the 4th European Conference on Software Engineering Education (ECSEE '20), Seeon/Bavaria, Germany, 18 June 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 12–21. [CrossRef]
54. Opt, S.; Sims, C.-D.L. Scrum: Enhancing Student Team Organization and Collaboration. *Commun. Teach.* **2014**, *29*, 55–62. [CrossRef]
55. Pope-Ruark, R. We Scrum Every Day: Using Scrum Project Management Framework for Group Projects. *Coll. Teach.* **2012**, *60*, 164–169. [CrossRef]
56. Chen, Z. Applying Scrum to Manage a Senior Capstone Project. *ASEE Annu. Conf. Expo.* **2017**. [CrossRef]
57. Schild, J.; Walter, R.; Masuch, M. ABC-Sprints: Adapting Scrum to academic game development courses. In Proceedings of the Fifth International Conference on the Foundations of Digital Games (FDG '10), Monterey, CA, USA, 19–21 June 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 187–194. [CrossRef]
58. Sanders, D. Using Scrum to manage student projects. *J. Comput. Small Coll.* **2007**, *23*, 79.
59. Linos, P.K.; Rybarczyk, R.; Partenheimer, N. Involving IT professionals in Scrum student teams: An empirical study on the impact of students' learning. In Proceedings of the 2020 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 21–24 October 2020; pp. 1–9. [CrossRef]

60. Paasivaara, M.; Lassenius, C.; Damian, D.; Räty, P.; Schröter, A. Teaching students global software engineering skills using distributed Scrum. In Proceedings of the 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, USA, 19 May 2013; pp. 1128–1137. [CrossRef]
61. Hidayati, A.; Budiardjo, E.K.; Purwandari, B. Hard and Soft Skills for Scrum Global Software Development Teams. In Proceedings of the 3rd International Conference on Software Engineering and Information Management (ICSIM '20), Sydney, Australia, 12–15 January 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 110–114. [CrossRef]
62. Garcia, I.; Pacheco, C.; Méndez, F.; Calvo-Manzano, J.A. The effects of game-based learning in the acquisition of "soft skills" on undergraduate software engineering courses: A systematic literature review. *Comput. Appl. Eng. Educ.* **2020**, *28*, 1327–1354. [CrossRef]