Towards a New Metamodel Approach of Scrum, XP and Ignite Methods

Merzouk Soukaina, Elkhalyly Badr, Marzak Abdelaziz, Sael Nawal
Department of Mathematics and Computer Sciences, Hassan II University- Casablanca
Faculty of Sciences Ben M'sik, Casablanca, Morocco

***Abstract*—The agile approach is a philosophy that aims to avoid the traditional management approach problems. It concentrates on the collaborative approach, using iterative and incremental development. The client receives a first production version (increment) of his software, faster thanks to agile methodologies. Project needs are influenced by the rapid expansion of technologies, particularly after the emergence of the Internet of Things (IoT). They are becoming larger and more complex. IoT provides a standardization and unification of electronic identities, digital entities, and physical objects. Consequently, interconnected devices can retrieve, store, send, and process data easier from both physical and virtual worlds. Scalable methods such as SAFe, LeSS, SPS, and others are existing methodologies ameliorated and dedicated to large projects. These methods are tough to adopt and do not consider the physical side of the project, according to IoT enterprise teams. Based on their managerial and IoT expertise, they suggest their own methods (Ignite | IoT Methodology and IoT Methodology). Model Driven Architecture (MDA) was coined by the Object Management Group (OMG) in 2000 to develop perpetual models that are independent of the technical intricacies of the execution platforms. The purpose of this paper is to propose a metamodel for each methodology among: Scrum, XP, and Ignite.**

Keywords*—*Agile software development; scrum; extreme programming; XP; internet of things; IoT; Ignite | IoT Methodology; IoT Methodology; metamodel; MDA; OMG

I. Introduction

For decades, projects have been managed with the classic (or traditional or predictive) approach, which is characterized by gathering the requirements, defining the product, developing, and testing it before the delivery. This is the "waterfall" model [1] or its adaptation, the "V" model [2].

One of the main weaknesses of 'waterfall' approach is that the design errors are often not discovered until the time of deployment. At this time, the project is almost complete, and errors are often costly to recover.

Agile methods avoid this weakness by executing iterative and incremental development that is carried out in a collaborative spirit, with the right amount of formalism. They generate a high-quality product while considering the modification needs of the customers.

Thanks to agile methods, the client participates in the realization of the project (prioritize, select items to be implemented on current iteration, do the functional tests, etc.) and obtains very quickly a first production release of his software, by using one of these methods: the XP method, the SCRUM method, the DSDM method, the FDD method, etc. [3].

Projects are becoming larger and complicated as the technology industry expands, especially after the emergence of the Internet of Things. The latter is defined as a network of interconnected electronic devices, which enables electronic identities, digital entities, and physical objects to be standardized and united. As a result, being able to recover, store, transmit, and process the associated data without interruption across the physical and virtual worlds [4].

The technology evolution leads project management experts to try different management methods or to improve existing ones. SAFe [5] [6], LeSS [7], SPS [8] and others are among the methods dedicated to large projects. IoT experts find that these methods, despite being dedicated to large and complex projects, they are complex in use and do not address the physical part of the project. At this level, IoT companies' teams propose their own methodologies based on their managerial and IoT experience. These methodologies are Ignite | IoT methodology (Ignite) [9] and IoT Methodology [10].

In 2000, the Object Management Group (OMG) coined the term Model Driven Architecture (MDA) to create perennial models that are independent of the technical minutiae of the execution platforms. This approach necessitates the use of a variety of models, including CIM, PIM, PSM, and others. As a result, the various formalisms that enable the building of models that are both sustainable and productive had to be explicitly specified. The MetaObject Facility (MOF [16]) standard, which was designed by OMG specifically for this purpose, supports the establishment of modeling formalisms in the form of metamodels. These are made up of a collection of metaclasses linked together through meta-association [11].

This article aims to present first of all the Scrum, XP and Ignite methodologies, describing their processes, artifacts, and roles. Then, it proposes a metamodel for each of these methods using MOF standards.

The rest of the paper is organized as follows: Section 2 presents the research work related to the field of project management and model engineering. Section 3 describes the methodology followed in this work. Section 4 is specific to the definition of the metamodel principle, the presentation of the proposed metamodel, and finally the description of the method

components that are the basis of the proposal. Section 5 presents the paper's discussion and conclusion.

II. Related Work

During this work, there are research works related to the metamodels of project management methods.

Many companies that use agile processes might benefit from using a process measurement framework, for example, to evaluate their process maturity. Ernesto et al. [12] offer a metamodel for the construction of specialized data models for agile development processes in their study. Then, they demonstrate how their metamodel can be utilized to derive a Scrum process model.

The traditional approaches of software development (e.g., RUP, waterfall) and agile approaches (e.g., Scrum, XP) are the two most popular software development strategies nowadays. Hybrid methods can also be used because both approaches offer advantages. Darko and Zeljko's work [13] demonstrates how to use metamodels to create new hybrid software engineering methodologies. They build a common metamodel by combining the metamodels of the traditional waterfall method with extreme programming. The new hybrid method development and development workflow are then built on top of this shared metamodel.

Given the rapid advancement of technology, the necessity for project management in terms of methodology and new concepts continues to develop. Hamzan and Belangour [14] constructed a framework for generating a metamodel that they used to project management to provide a generic metamodel of project management. This approach is founded on two project management methodologies which are "PRINCE 2" and "Scrum". The goal of this research is to validate and apply this methodology to all aspects of IT governance, then merge the metamodels to build a global metamodel that covers all IT governance domains.

The Agile Project Management Framework (APMF) is a collection of fine-grained project management techniques used in agile methodologies that is quickly gaining traction as a viable alternative to traditional project management frameworks. However, both frameworks have flaws that prevent developers from improving one in order to accept the other. Merging the two systems into a Unified Project Management Framework (UPMF) is a reasonable option. In order to achieve this goal, Mahsa and Raman [15] propose a project-management technique metamodel as a common abstract substrate for fusing the traditional framework with its agile counterpart. By abstracting the fine-grained parts of APMF, the proposed Agile Project Management Methodology Metamodel (APM3) was created. An analytical analysis of the project management procedures of seven important agile approaches was undertaken using APM3's generic agile metamodel.

A standardization of the methods concerning Scrum, XP and Ignite is proposed. This standardization is a metamodeling of the phases, the artifacts, and the whole ecosystem.

III. Methodology

The carrying out of the work is done by applying the Scrum methodology as shown in the Fig. 1. The first step consists in defining a list of tasks in the form of user stories that should be carried out throughout the work and to refine them. This list is not definitive, as it will be updated during the work. The next step is prioritizing the list according to the importance of the user stories and its sequence. The second step is to select the user stories to be done for each sprint. Finally, an increment is produced, and the next sprint is started, and so on.

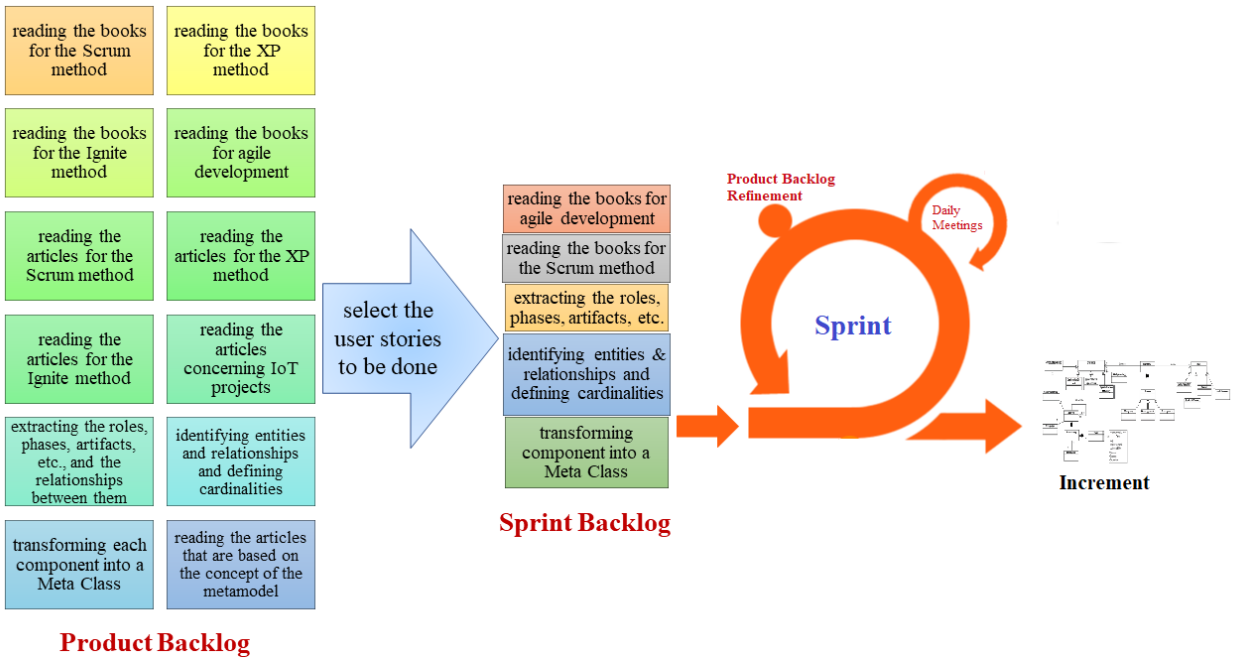

Fig. 1. The Work using Scrum Process.

The list of user stories contains: reading the books for the Scrum method; reading the books for the XP method; reading the books for the Ignite method; reading the books for agile development; reading the articles for the Scrum method; reading the articles for the XP method; reading the articles for the Ignite method; reading the articles concerning IoT projects; extracting the roles, phases, artifacts, etc., and the relationships between them; identifying entities and relationships and defining cardinalities; transforming each component into a Meta Class; reading the articles that are based on the concept of the metamodel.

- The first Sprint concerns the realization of the Scrum metamodel.
- The 2nd Sprint concerns the realization of the XP metamodel.
- The 3rd Sprint concerns the realization of the Ignite metamodel.
- The 4th Sprint concerning the proofreading, writing and correction of the proposed metamodels.

Moreover, daily meetings are held for the discussion regarding the work and the problems encountered in the realization of the current activity.

IV. Metamodel Approach

A. Definition

The Object Management Group (OMG) defined Model Driven Architecture (MDA) in 2000.

This approach advocates the massive use of models and offers the first answers to how, when, what and why to model. It includes the definition of several standards, notably UML, MOF, and XMI. The main objective of MDA is the development of perennial models, independent of the technical details of the execution platforms, in order to allow the automatic generation of the entire application code and to obtain a significant gain in productivity.

MDA necessitated the employment of a variety of models. As a result, it was necessary to explicitly specify the many formalisms that permit the construction of models that are both sustainable and productive. The MOF [16] standard, created by OMG for this purpose, provides support for establishing modeling formalisms in the form of metamodels. According to MOF, any model must respect the structure defined by its metamodel. A metamodel is thus composed of a set of metaclasses. The latter has a name and contains attributes and operations, also called meta-attributes and meta-operations. A meta-association is a binary association between two metaclasses. A meta-association has a name, and each of its ends can have a role name and a multiplicity [11].

B. Extreme Programming Methodology

1) Definition: Extreme programming, or XP, is a method proposed by Kent Beck that applies the old development principles to the extreme. It divides the project into subprojects applying the traditional development steps in each subproject in an iterative way and continuous integration (incremental) which reduces the change cost [17] [18] [19] [20]. Fig. 2 shows the evolution of the Waterfall model towards extreme programming.

2) XP Metamodel: Fig. 3 shows the metamodel of the XP method. This metamodel is based on the transformation of the method's components into metaclasses and the relationships between them into meta-associations.

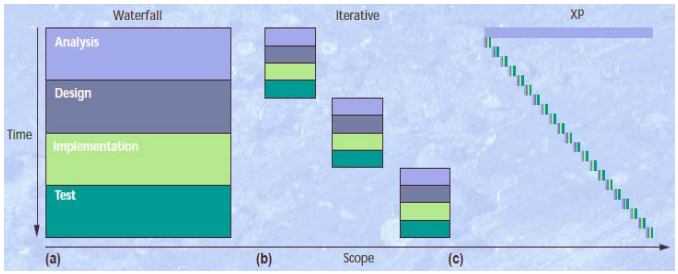

Fig. 2. Evolution towards XP Method: (a) Waterfall Model, (b) Spiral Model, (c) XP [18].

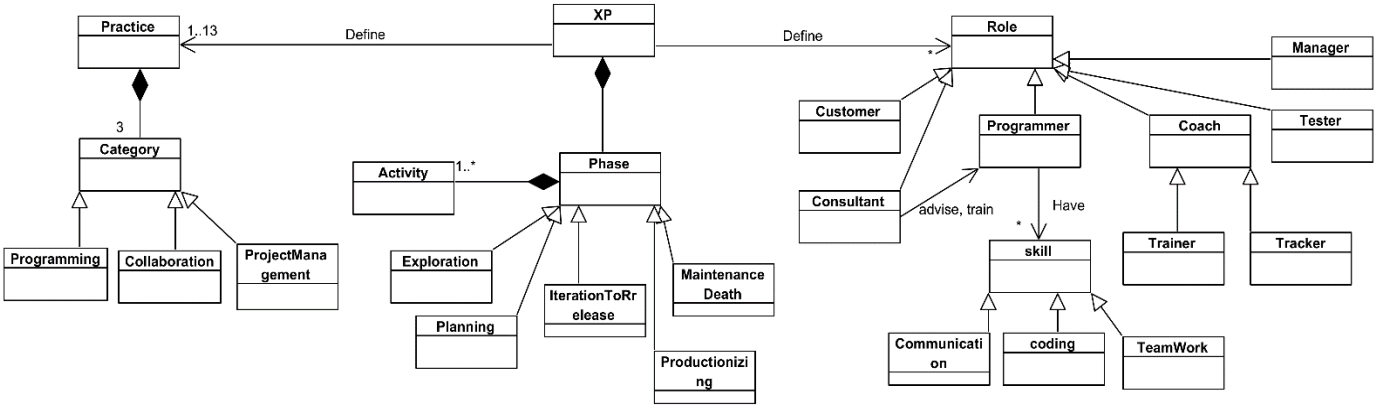

Fig. 3. Proposed Metamodel for XP Method.

3) Component:

a) Phases: The XP method consists of five phases viz., exploration, planning, iteration to release, productionizing, maintenance, and death. Each phase consists of a set of activities. These phases are described according to [20], [21], and [22].

- Exploration: The customer uses index cards to write user stories presenting his needs, as shown in the Fig. 4. These user stories are estimated by developers one by one in terms of the time needed to implement them and the implementation risk. This phase should take a few weeks to a few months and defines the technologies, tools and architectures that will be used in the project. Fig. 4 shows an example of a user story card used in the C3 project.

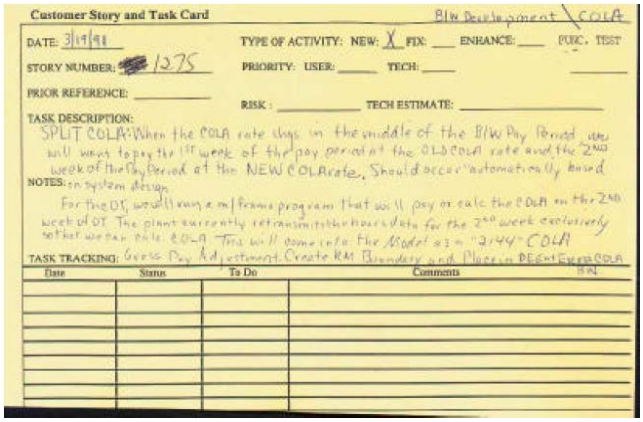

Fig. 4. C3 Project user Story Card [21].

- Planning: Commitment schedule meeting is done, after all user stories are written by the customer and estimated by the developers, to define the priority of each story and which ones are needed for the current release. In addition, the customer writes the functional tests based on the user story cards. The team transforms these cards into tasks with an estimate of each (the Fig. 5 shows a task card used in the C3 project). The meeting ends when the list of stories and the schedule are validated.

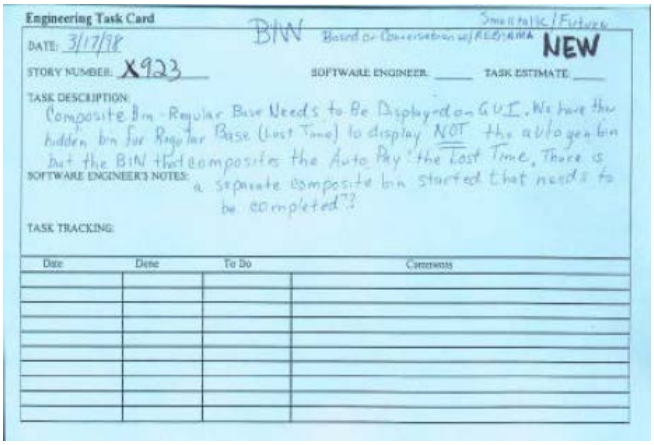

Fig. 5. C3 Project Engineering Task Card [21].

- Iteration To Release: This phase consists of breaking down the schedule of commitments set in the previous phase in a series of iterations. Each iteration follows the phases of the classical approach (designing, coding, testing, and integration). Furthermore, functional tests are applied at the end of each iteration to verify the functioning of the story.

- Productionizing: The system is ready for production at the end of the last iteration. In this case, it is necessary to ensure the performance of the system before delivering it to the customer. To do this, extra testing is done. The ideas and proposals reported are documented for later implementation during, for example, the maintenance phase.

- Maintenance and Death: The maintenance phase is triggered after the first release to the customer. The team must keep the system running in production while the new iteration is in production. It may also require the integration of new people into the team and the modification of the team structure. The death phase is the phase that describes the end of the project when the customer is satisfied and has no stories to implement. At this point, the system documentation is written. This phase also consists of closing down the system if it does not deliver the desired results or if it becomes too costly for further development.

b) Roles: Furthermore, XP defines six roles viz., Customer, Consultant, Programmer, Coach, Tester, and Manager that are described according to [18], [21], and [22]. The consultant is responsible for advising and training the programmer that having communication, coding, and teamwork skills.

- Customer: The client is responsible for defining the requirements because he writes the user stories. In addition, he defines the priority for each card and writes the functional tests which are used at the end of the iteration to check that the stories work. There is a special role in the XP method called on site customer. This is often a domain expert representing the customer.

- Consultant: In most XP projects, there are no professionals due to the rules and practices. A consultant will be employed in these circumstances to supply this information. The consultant's job is to provide expertise. One or two team members will meet with the consultant and ask several of the technical questions before attempting to fix the problem.

- Programmer: The programmer is responsible for the analysis of the design code, etc. He writes the program code as simple as possible. He is required to be competent in communication, coding, and teamwork.

- Managing part of XP project: The project management part of XP is divided into two roles such as coach and tracker Coach: The system manager participates in the management meetings. His role is to guide the team away from the process. It is necessary to be calm; to understand alternative practices that need it and could

help the current set of problems; what the ideas behind XP are; how it relates to the current situation and how other teams use XP. Tracker: Acts as the conscience of the team. He/she must track to determine if the Iteration Schedule and Commitment Schedule can be met. This work gives him data that is used to give feedback to developers on the quality of their estimates. Also, it helps him get feedback on the team's next estimates. In addition, the tracker is required to be proficient in collecting the necessary information without disrupting the whole process.

- Tester: The programmers have a large portion of the duty of testing, so the role of the tester in an XP team is particularly customer-centric. However, someone needs to run all the tests regularly, disseminate the test findings, and guarantee that the testing instruments are in good working order.
- Big Boss: Courage, confidence, and the occasional insistence that they do what they say they will be what the team most needs from the Big Boss. They aren't complaining; they genuinely aren't. They want Big Boss to know as soon as possible if things aren't going as planned, so he can react as quickly as feasible. If it works, he will be golden because he will have a team that's productive, satisfied with its clients, and does everything they can to avoid surprising him.

c) Practices: The method's practices are grouped into three categories, such as programming, collaboration and project management, which are described according to [21], [22], [23], and [24].

Programming category

- Simple design: The simplest solution that will work is always implemented by developers. They do not, for example, design generic mechanisms if the urgent necessity does not necessitate it.
- Refactoring: Developers are not hesitant to go back over the written code to make it "cleaner," to remove any unused components, and to prepare it for the addition of the next feature. More generally, this practice suggests a continuous design approach that highlights the application's structure as it develops.
- Test-first programming, unit tests, developer tests: Even as they are writing the code, developers generate automated tests for it. This enables them to gain a deeper understanding of the problem before creating the code. In addition, to gradually build up a battery of tests that enables them to make changes to the application fast and with confidence.
- Acceptance Tests, Customer Tests: Through participating in the writing of acceptance tests, the client expresses his wants and the programmers' objectives very explicitly. Acceptance tests, like unit tests, must be automated in order to ensure that the product does not regress on a daily basis.

Collaboration category

- Pair Programming: The developers always work in pairs on the same machine when coding for the application - this is an "extreme" type of code review that both developers actively collaborate to resolve issues they discover. The pairs change regularly, so everyone must work with all other team members early or later.
- Collective code ownership: All developers in the team may be required to work on all parts of the application. Furthermore, they have a duty to improve the code they work on, even if they are not the original authors.
- Coding standards: Developers follow coding rules defined by the team itself. This ensures that their code is consistent with the rest of the application, and therefore facilitates the intervention of other developers.
- Metaphor: Developers do not hesitate to use metaphors to describe the internal structure of the software or its functional issues. This facilitates communication and ensures a certain homogeneity of style throughout the design, the ideal being to describe the system in its entirety by a single metaphor.
- Continuous integration: Developers synchronize their work as frequently as feasible, at least once a day. This decreases the frequency and severity of integration issues, while also ensuring that a current version of the software is always available.

Project Management Category

- Frequent releases: The team delivers software releases at a regular rate, as high as possible, depending on the client's preferences. This enables both the team and the client to guarantee that the product meets the client's expectations and that the project remains on schedule.
- Planning game: In dedicated sessions done on a regular schedule throughout the project, the client and the development team collaborate on project planning.
- On-site customer, whole team: The customer is literally incorporated into the development team, allowing him to set priorities and specify his wants clearly, notably by directly addressing programmers' inquiries and taking advantage of the instant feedback provided by a frequent-delivery application.
- Sustainable pace: The team adopts schedules that allow it to keep the energy required to create high-quality work and efficiently follow other project procedures.

C. Scrum Methodology

1) Definition: The word "scrum" and the method's idea are derived from a rugby strategy that entails "bringing an out-of-play ball back into the game" through collaboration [25] [26]. The method was created in the early 1990s to manage the systems development process. It is a framework that focuses on how team members should work together and always ready to reorient so as to create a flexible system in an ever-changing environment. Scrum helps an organization's existing

engineering processes by requiring frequent management activities to systematically identify gaps or impediments in the development process, as well as the techniques that are employed [27] [19].

2) Scrum metamodel: The Fig. 6 shows the metamodel of the Scrum method. This metamodel is based on the transformation of the method's components into metaclasses and the relationships between them into meta-associations.

3) Component:

a) Phases: Scrum process consists mainly of three phases viz; Pre-Game, Development and Post-Game, which are described according to [20] and [22].

- Pre-Game: Pre-game consists of two sub-phases, the planning phase, which serves to define the system being developed. Defining the customer requirements and estimating the effort needed to implement each requirement. The list of requirements is always updated with new requirements. This phase ends with the definition of the project team, tools, and resources. The architecture phase consists of the high-level design of the system based on the requirements determined in the previous phase and the preparation of preliminary plans for the content of the releases.
- Development: The development phase or game phase is a black box where the system is developed in sprints that comprise the traditional software development phases. In addition, Scrum identifies environmental and technical variables. Then, aims to control them through various Scrum practices during the sprint, which is planned to take from one week to one month.
- Post-Game: The closing phase of the release after the customer has reached his goal, and he has no change or other requirements. Release preparation includes the integration, testing, and documentation of the final system.

b) Roles: Scrum defines the roles, described according to [28], which are Scrum Master, Product Owner, Team, and Stakeholders.

- Scrum Master: He is the team leader and at the same time a team member, he helps the team to understand the Scrum methodology, to create a high-value increment. He ensures that there are no obstacles that stop the progress of the product and that the team respects the work schedule. It helps the product owner to define and manage the product backlog. Then, facilitates the collaboration of stakeholders according to the demands or needs and to remove the barriers between them and the team [27].
- Product Owner: This is the most important role in this method. For the reason that it is the representative of the customer and responsible for defining the product vision and following its transition to the product backlog list. The latter is used by the product owner to check that the requirements are developed. The most significant skill that the product owner is supposed to have is written and oral communication [29].
- Team: Responsible for the project from planning to delivery of an increment. It is self-organized, works together, and takes account of the probability of change in requirements.
- Stakeholder (s): It is a person or a group of people or organizations that have a relationship with the project who are the clients [30].

c) Practices: It also defines seven practices divided into two categories, events, and artifacts to be applied in different phases to avoid chaos caused by the unpredictability and complexity. The practices, which are described according to [22], [27], and [31], are: Daily Meeting, Sprint Planning Meeting, Sprint Review Meeting, Effort Estimation, Sprint, Sprint Backlog, Product Backlog. Sprint has a Velocity based on the Sprint Backlog, which is produced from the Product Backlog. In addition, it consists of a set of User Stories that contain Elements and contain Tasks of different types.

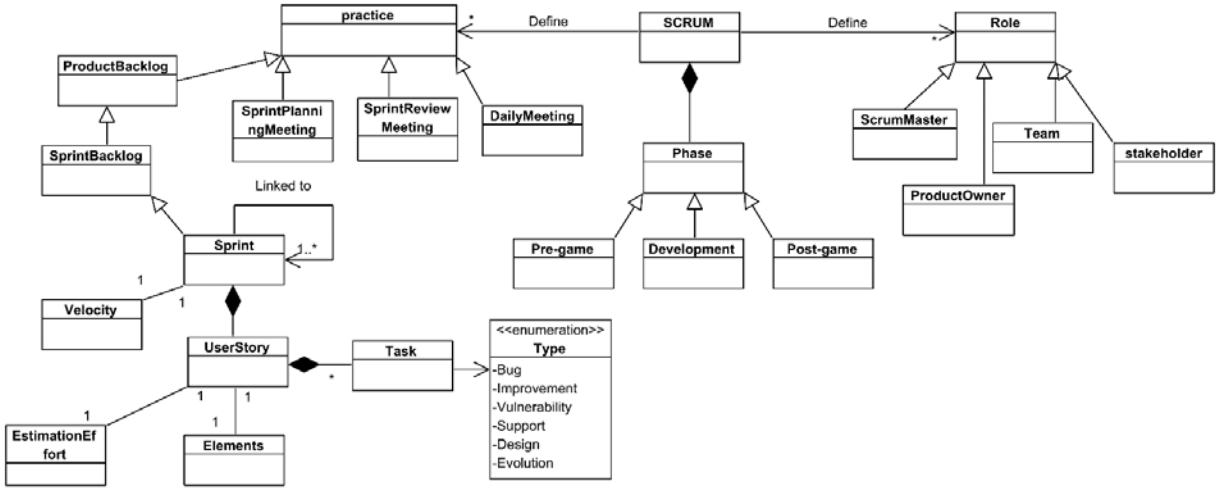

Fig. 6. Proposed Metamodel for Scrum Method.

Events category

- Daily Meetings: Scrum meetings are led by the Scrum Master. In addition to the Scrum team, management can also attend the meeting. The Scrum Team's Developers attend the Daily Scrum, which lasts 15 minutes. Developers are those who are actively working on Sprint Backlog items, such as the Product Owner or Scrum Master. Daily Scrums improve communication, identify barriers, stimulate quick decision-making, and thereby avoid the need for additional meetings.
- Sprint Planning Meeting: The Scrum Master organizes a Sprint Planning Meeting that is a two-section meeting. In the primary section of the meeting, customers, users, management, the Product Owner, and the Scrum Team determine the objectives and functionalities of the subsequent Sprint. The Scrum Master and the Scrum Team hold the second section of the meeting, which focuses on how the product increment is implemented in the course of the Sprint.
- Sprint Review Meeting: The Sprint Review's goal is to examine the Sprint's results and make recommendations for future changes. In an informal meeting, the Scrum Team and Scrum Master present the outcomes of their work to management, customers, users, and the Product Owner. The participants evaluate the product increment and make decisions about the next steps. The review meeting may result in the addition of new Backlog items and possibly a change in the system's direction.

Artifact's category

- Sprint: The procedure of adjusting to changing environmental variables is known as sprint (requirements, time, resources, knowledge, technology, etc.). In a Sprint, where ideas are becoming valuable, the Scrum Team arranges itself to generate a new executable product increment over the course of thirty calendar days. Sprint Planning Meetings, Sprint Backlog, and Daily Scrum meetings are the team's working tools. Each Sprint may be in concept of as a small project. Burndowns, burn-ups, and cumulative flows are all techniques for forecasting progress. While they have proven to be valuable, they do not take the place of empirical evidence. What's going to occur in complicated environments is unknown. Only what has already occurred can be used to make decisions in the future. If the Sprint Goal is no longer relevant, the Sprint may be cancelled. Only the Product Owner has the right to terminate the Sprint. During the Sprint, no modifications are made that might endanger the Sprint Goal; quality is maintained; the Product Backlog is adjusted as required; and, as additional information becomes available, with the Product Owner, the Scope can be clarified and renegotiated.
- Sprint Backlog: A Sprint Backlog is created at the start of each Sprint. The Sprint Backlog is made up of the Sprint Goal (why), the Product Backlog items chosen for the Sprint (what), and an actionable plan for producing the Increment (how). The Sprint Backlog is a strategy created by and for developers. The items are chosen in the Sprint Planning meeting by the Scrum Team, in collaboration with the Scrum Master and the Product Owner. Which are based on prioritized items and Sprint Goals. The Sprint Backlog, in contrast to the Product Backlog, is stable until the Sprint is finished. A new iteration of the system is deployed once all the items in the Sprint Backlog have been accomplished.
- Product Backlog: Based on existing knowledge, the Product Backlog describes everything that is required in the final product. As a result, the Product Backlog describes the project's tasks. Features, functionality, bug fixes, issues, requested improvements, and technology upgrades are all examples of backlog items. The list also includes issues that must be resolved before other Backlog items may be completed. Product Backlog items can be created by a variety of actors, including the customer, project team, marketing and sales, management, and customer support. The Product Owner oversees keeping the Product Backlog updated.
- Effort Estimation: The Product Owner and the Scrum Team oversee effort estimation, which is an iterative procedure. The Product Owner collaborates with others as the backlog is generated to predict how long it will take to develop. He or she consults with developers, technical writers, quality assurance staff, and others who are familiar with the product and technology in order to arrive at the estimate. Because the product owner and the team are experienced in estimating, the estimate will be accurate. The Product Owner develops estimates for every item, beginning with the highest priority backlog.

D. Ignite Methodology

1) Definition: Originally from industry. The founders of this methodology are based on the analysis of manufacturing and industry, connected vehicles, Smart Energy and Smart Cities. Through collecting best practices of IoT strategy management and project execution. It is open source and covers all aspects of IoT development. It addresses various IoT stakeholders, namely product managers, project managers, and solution architects [9] [32] [33] [34] [35].

2) Ignite metamodel: Fig. 7 shows the metamodel of the Ignite method. This metamodel is based on the transformation of the method's components into metaclasses and the relationships between them into meta-associations.

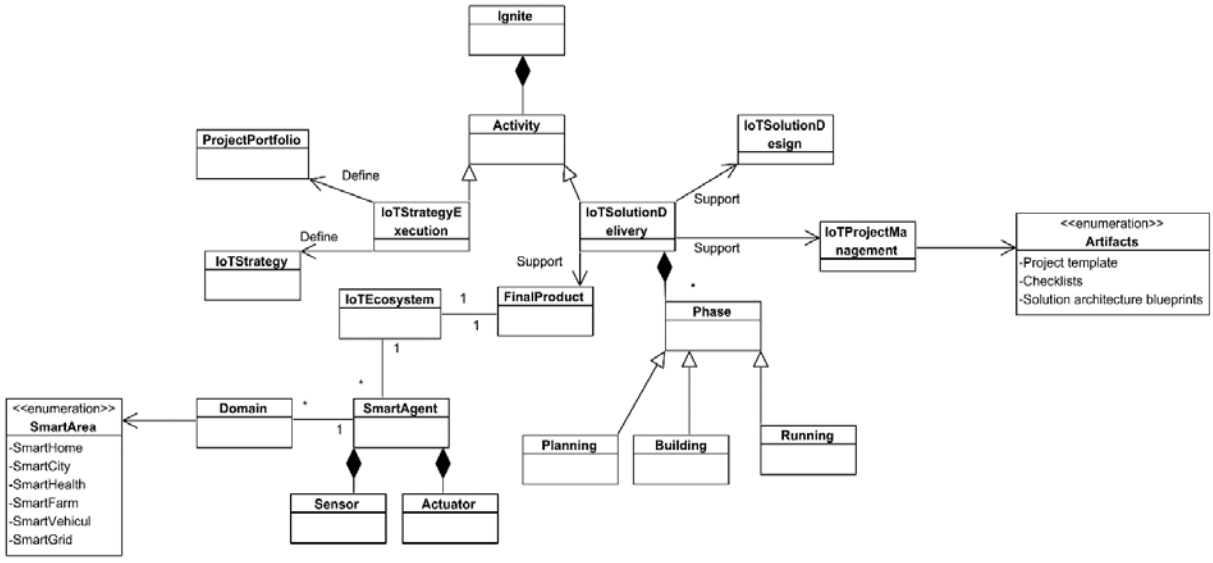

Fig. 7. Proposed Metamodel for Scrum Method.

3) Component: All components of this methodology are described according to [9].

a) Phases: Two activities make up the Ignite process (as shown in Fig. 8). The first is to define the strategy and prepare the organization to adopt IoT, then create and manage the portfolio of IoT projects to support the IoT Strategy. This activity is called IoT Strategy Execution. The second activity, called IoT Solution Delivery, is used to execute the famous three phases such as Plan, Build and Run to deliver a solution.

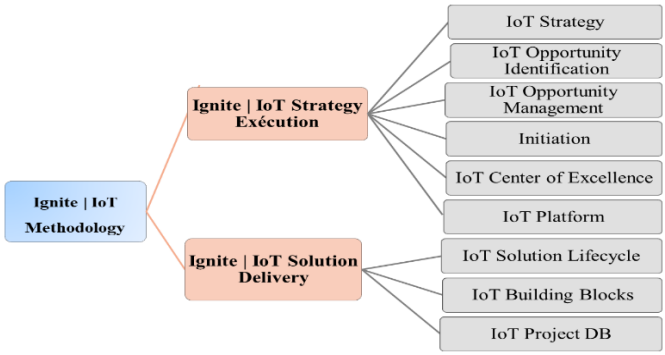

Fig. 8. Ignite | IoT Methodology Activities.

Ignite | IoT Strategy Execution

IoT Strategy, IoT Opportunity Identification, IoT Opportunity Management, Initiation, IoT Center of Excellence, and IoT Platform are the six domains of the Ignite | IoT Strategy Execution framework.

- IoT Strategy: The extent and speed with which a corporation should migrate toward IoT should be reflected in its IoT strategy. The Internet of Things strategy must have a vision, goals, and guiding principles. It should also provide a high-level overview of how IoT-related business areas should create strategic alliances and partner ecosystems. Finally, it must oversee the portfolio of IoT prospects and projects, as well as budgeting and IoT roadmap management.
- IoT Opportunity Identification: The generation of IoT solution innovation ideas, can take two forms: an open process that taps into the creativity of employees, customers, and developers, or a more structured approach in which ideas are derived from a specific context, such as the company's value chain. Ideas that show the most promise should be fleshed out further, perhaps using idea refinement templates.
- IoT Opportunity Management: The most promising ideas are then improved as part of the IoT Opportunity Management process after passing the first quality gate. In order to examine the utility and the business case, a more complete business model must be created. The following Impact & Risk Assessment step guarantees that all conceivable results of the business model are taken into account.
- Initiation: An IoT opportunity can be moved to the Initiation stage once it has been authorized. Management must decide how to best set up the effort at this stage, e.g., as a dedicated internal project, a spin-off, or even an M&A project. These activities connect to the Ignite | IoT Solution Delivery for internal initiatives.
- IoT Center of Excellence: An IoT Center of Excellence (CoE) can assist new projects in gaining traction faster. For instance, by offering IoT consulting and alter management support, IoT maturity evaluations can assist a company in determining where it stands in terms of IoT adoption.
- IoT Platform: Large enterprises may find it beneficial to provide a shared IoT Platform that many projects can use to create their solutions. An IoT application platform, connectivity solutions, and technical and functional standards are typically included.

Process

- Generate Idea: In a large company, there are usually two approaches to produce ideas: open idea generation (green field technique) or a more structured idea generation approach. The latter approach is generally carried out in a top-down manner. It typically entails a thorough market investigation by an internal strategy team or an external consulting agency. Open idea creating is more likely to produce disruptive ideas. As a result, companies should have numerous channels in place to collect these ideas, including employees, consumers, and even developers.
- Refine Idea: Many good ideas aren't particularly attractive when they're initially formed. Before they will really persuade potential stakeholders, they need care in order to grow and mature. Thankfully, there is no shortage of ideation methodologies that promote idea refining, such as the St. Gallen Business Model Navigator™ and the Innovation Project Canvas. The detailed idea sketch, which is the product of the idea-refinement phase, can be used for presentations at the next quality gate level. It can be used to develop the business model after it has been approved.
- Business Model Development: it consists of four phases (shown in the Fig. 9) viz., Strategic embedding (it lays the foundations of the business model and ensures consistency with the IoT strategy or the company's IoT vision. The implementation of "future proofing" should indicate how the business model intends to address future challenges.); Value proposition (To increase the attractiveness of the offer for customers, the proven approach of segmentation of target groups, formulation of the value offer and definition of customer channels can be used.); Customer journey (The explanation of the end-to-end solution from the customer's perspective serves in highlighting the characteristics of the proposal that the consumer finds important. Another benefit of establishing the customer journey is that it guarantees that all relevant consumer channels have been identified.); Value added (The value added can be demonstrated once the solution has been defined. The capabilities of the parties are the network's constituent elements: they are a combination of technology, resources, and know-how that they can bring in to assist the solution.); Business case (There are numerous techniques and templates for calculating business cases, but the recommendation is to use the same one for all IoT activities, as this makes comparing business models easier.); Strategic impact and subsequent business models (The house's chimneys represent two non-monetary effects of a business model that must be considered alongside the business case. The second chimney, “subsequent business models” is extremely specific to the IoT: it is very usual for teams to come up with exciting new ideas on how to utilize the data. Additionally, build new services while designing the business model and gathering all the associated data with connected devices.).

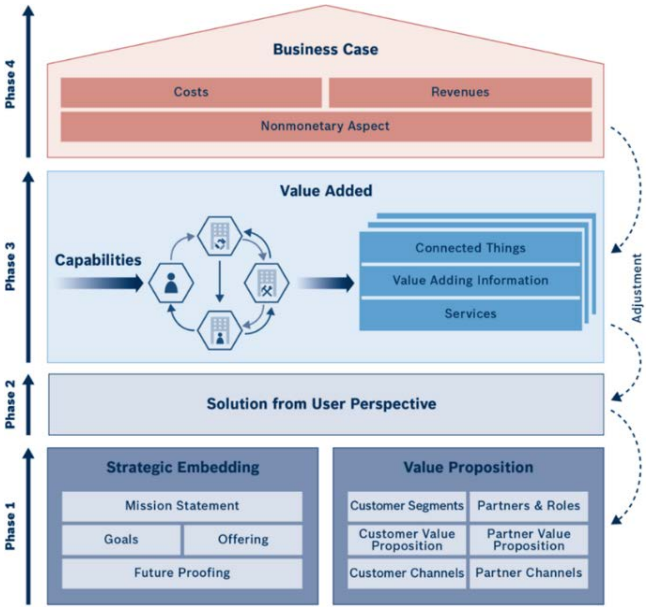

Fig. 9. Builder of IoT Business Model [9].

- Impact And Risk Analysis: Business models, and business cases in particular, deal with future value flows, so they are subject to uncertainty. To promote transparency for decision-makers and to generate the tasks necessary to resolve these uncertainties, it is critical to underline the degree of uncertainty within the business model. Various future scenarios are suggested that support the provided parameters. Trends or cause-and-effect relationships can be used to accomplish this. In the context of the strategy, it is crucial to check those aspects of the business model that create effect and value.

2 Ignite | IoT Solution Delivery

- By providing project templates, checklists, and solution architectural blueprints, the mission is to make IoT best practice applicable in the form of a technology-independent, reusable, open-source methodology that supports IoT solution design as well as the implementation and management of IoT projects. The following is a breakdown of Ignite | IoT Solution Delivery.
- The IoT Solution Lifecycle focuses on the planning, development, and execution of IoT solutions encloses the following elements. Initial Project Design: The elements established as part of the generic IoT Building Blocks, such as project self-assessment employing IoT Project Dimensions, solution architecture employing IoT Architecture Blueprints, and technology selection employing IoT Technology Profiles, are all used in this design blueprint. Project workstreams and project organization: The top-level organization and workstreams generally found in an IoT solution project are defined in this blueprint. There is a checklist for every workstream, as well as a list of common dependencies between them.

- IoT Building Blocks includes reusable artifacts such as IoT Project Dimensions, IoT Architecture Blueprints, and IoT Technology Profiles from successful projects.
- IoT Project DB is a repository of reference projects that have been analyzed in order to identify best practices for the IoT Solution Lifecycle and Building Blocks.

Process

- IoT Project Initiation: A requirements study, which is more in-depth than the analysis performed during the business model building phase, is a significant factor in the Ignite | IoT Project Initiation phase. A tiny team of subject matter specialists is generally in charge of project initiation. A business analyst with strong domain expertise and a clear vision for the solution's functional features should also be part of the team.
- Initial Solution Design: Initial Solution Design consists of a collection of key artifacts that include analysis, projections, and planning, as well as functional and technical design artifacts. Even though they might be developed concurrently, it is generally more practical to group them, as shown the Fig. 10. Analysis, Projections, Planning: It was created to aid with analysis, projections, and planning. They contain: Problem Statement, Stakeholder Analysis; Site Survey; Solution Sketch; Project Dimensions; Quantity Structure; Milestone Plan.

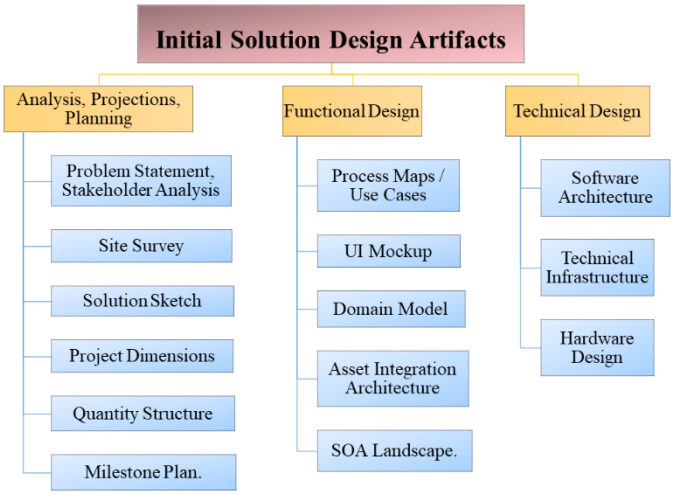

Fig. 10. Initial Solution Design Artifacts.

Functional Design contains: Process Maps / Use Cases; UI Mockup; Domain Model; Asset Integration Architecture; SOA Landscape. Technical Design contains: Software Architecture; Technical Infrastructure; Hardware Design.

- Plan: After the funding decision on Ignite | IoT Strategy Execution, this phase begins. A small, but devoted team usually creates an initial project plan, which includes a solution definition, based on the ideas and criteria from the business planning phase. This could be an RFP (Request for Proposal) document or the initial list of high-level epics that will need to be broken down into more detailed user stories later in the Build phase. The initial team will often oversee sourcing (internally or externally) the larger team that will eventually create the solution during the planning phase.
- Build: A larger team or teams often execute the build phase. Keep in mind that, particularly in the IoT, the work is frequently with various, multidisciplinary teams. It's worth noting that, due to the often highly dynamic nature of IoT projects, planning continues during the build phase. Each sprint will be meticulously planned, especially if an Agile approach is used. The higher-level papers developed during the planning phase will frequently need to be updated to reflect new or changing needs, as well as lessons gained from prior sprints.
- Run: The project team is typically disbanded, and the solution is handed over to a line organization around the time of the IoT solution's Start of Production (SOP). This line organization will set up an integrated DevOps organization in modern enterprises, which will deal with both the solution's continuous development and operations. DevOps for IoT can be more challenging than typical DevOps due to the potentially extremely distributed nature of IoT systems.

V. Discussion and Conclusion

The previous section presents the Scrum and XP agile methods and the dedicated IoT project method Ignite. Moreover, it presents the metamodel of each one with their components that have been translated into metaclasses and meta-associations.

The Scrum and XP methods are based on almost the same principles, with a very clear definition of roles, unlike the Ignite method.

Furthermore, another difference between Ignite and the Scrum and XP methods is that Ignite divides the project realization process into two sub-processes called Strategy Execution and Solution Delivery activity, whereas Scrum and XP have an ecosystem whose components are chained.

To sum up, the paper presents the standardization of the Scrum, XP and Ignite methods as metamodels based on their components and the fundamental MDA principles. These metamodels are the beginning of the forthcoming contribution concerning a Framework used for Industry 4.0.

References

[1] L. Sherrell, “Waterfall Model,” in Encyclopedia of Sciences and Religions, A. L. C. Runehov and L. Oviedo, Eds. Dordrecht: Springer Netherlands, 2013, pp. 2343–2344. doi: 10.1007/978-1-4020-8265-8_200285.

[2] I. Graessler, J. Hentze, and T. Bruckmann, “V-MODELS FOR INTERDISCIPLINARY SYSTEMS ENGINEERING,” 2018, pp. 747–756. doi: 10.21278/idc.2018.0333.

[3] S. Merzouk, S. Elhadi, A. Cherkaoui, A. Marzak, and N. Sael, “Agile Software Development: Comparative Study,” SSRN Electron. J., 2018, doi: 10.2139/ssrn.3186323.

[4] A. Abddessamad, S. Cherkaoui, M. Sm, A. Abdelaziz, M. Marzak, and H. Mh, “Review on Embedded Systems and the Internet of Things: Comparative Study,” Assoc. Comput. Mach., p. 7, 2021, doi: 10.1145/3454127.3457636.

[5] R. Knaster, SAFe 4.0 distilled: applying the Scaled Agile Framework for Lean software and systems engineering. Boston, MA: Addison-Wesley, 2017.

[6] "What is SAFe | Scaled Agile," SAFe® Enterprise Solutions. https://www.scaledagile.com/enterprise-solutions/what-is-safe/

[7] C. Larman, B. Vodde, and B. Jensen, Large-scale scrum. Dpunkt, 2017.

[8] G. Verheyen, "Scaled Professional Scrum – NexusTM," p. 5.

[9] D. Slama, F. Puhlmann, J. Morrish, and R. M. Bhatnagar, Eds., Enterprise IoT: Enterprise IoT: strategies and best practices for connected products and services. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly, 2016.

[10] "IoT Methodology – The Internet of Things project lifecycle guide for creative, technical and business people." http://www.iotmethodology.com/ (accessed May 30, 2020).

[11] X. Blanc and O. Salvatori, « MDA in action model-driven software engineering », MDA en action ingénierie logicielle guidée par les modèles. Paris: Eyrolles, 2005. [Online]. Available: https://www.eyrolles.com/Informatique/Livre/mda-en-action-9782212115390/

[12] E. Damiani, A. Colombo, F. Frati, and C. Bellettini, "A Metamodel for Modeling and Measuring Scrum Development Process," in Agile Processes in Software Engineering and Extreme Programming, vol. 4536, G. Concas, E. Damiani, M. Scotto, and G. Succi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 74–83. doi: 10.1007/978-3-540-73101-6_11.

[13] D. Androcec and Z. Dobrovic, "Creating Hybrid Software Engineering Methods by Means of Metamodels," p. 6.

[14] H. Ibrahim and B. Abdessamad, "Project Management Metamodel Construction Regarding IT Departments," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 10, 2019, doi: 10.14569/IJACSA.2019.0101029.

[15] M. H. Sadi and R. Ramsin, "APM3: A Methodology Metamodel for Agile Project Management.," 2009, pp. 367–378.

[16] "MetaObject Facility | Object Management Group." https://www.omg.org/mof/ (accessed Nov. 27, 2021).

[17] A. Anderson, R. Beattie, and K. Beck, "Chrysler goes to extremes," Distrib. Comput., 1998.

[18] K. Beck, "Embracing change with extreme programming," Computer, vol. 32, no. 10, pp. 70–77, Oct. 1999, doi: 10.1109/2.796139.

[19] S. Merzouk, "A Comparative Study of Agile Methods: Towards a New Model-based Method," vol. 9, no. 4, p. 8, 2017.

[20] S. Merzouk, A. Cherkaoui, A. Marzak, N. Sael, and F.-Z. Guerss, "The proposition of Process flow model for Scrum and eXtreme Programming," Assoc. Comput. Mach., p. 6, 2021, doi: 10.1145/3454127.3457627.

[21] T. Dudziak, "Extreme programming an overview," Methoden Werkzeuge Softwareproduktion WS, vol. 1999, pp. 1–28, 2000.

[22] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," 2002.

[23] K. Beck, Extreme programming explained: embrace change. addison-wesley professional, 2000.

[24] Jean-Louis Bénard, Laurent Bossavit, Régis Médina, and Dominic Williams, "EXtreme Programming: project management", EXtreme Programming: gestion de projet. Paris: Eyrolles, 2004. [Online]. Available: https://www.eyrolles.com/Informatique/Livre/gestion-de-projet-extreme-programming-2eme-tirage-2005-9782212115611/

[25] K. Schwaber, "SCRUM Development Process," in Business Object Design and Implementation, J. Sutherland, C. Casanave, J. Miller, P. Patel, and G. Hollowell, Eds. London: Springer London, 1997, pp. 117–134. doi: 10.1007/978-1-4471-0947-1_11.

[26] J. Sutherland and K. Schwaber, "Nut, Bolts, and Origins of an Agile Framework," p. 224.

[27] K. Schwaber and J. Sutherland, "The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game," Nov. 2020, [Online]. Available: https://scrumguides.org/scrum-guide.html

[28] S. A. Shinde, "Analysis of Agile Project Management with Scrum Method and Extreme Programming," IJSETR, vol. 4, p. 7, May 2015.

[29] S. Oomen, B. De Waal, A. Albertin, and P. Ravesteyn, "How can Scrum be succesful? Competences of the scrum product owner," 2017.

[30] A. Pham and P. V. Pham, Scrum in action: agile software project management and development. Boston: Course Technology PTR, 2012.

[31] K. Schwaber and M. Beedle, Agile Software Development with Scrum, Illustrée., vol. 1. Prentice Hall, 2002. [Online]. Available: https://books.google.co.ma/books?id=BpFYAAAAYAAJ

[32] I. Jacobson, I. Spence, and P.-W. Ng, "Is There a Single Method for the Internet of Things?," Queue, vol. 15, no. 3, pp. 25–51, 2017, doi: https://doi.org/10.1145/3106637.

[33] G. Görkem, T. Bedir, and T. Eray, "IoT System Development Methods," in Internet of things challenges, advances, and applications, Chapman & Hall/CRC Press, 2018, pp. 141–159.

[34] S. Merzouk, A. Cherkaoui, A. Marzak, and S. Nawal, "IoT methodologies: comparative study," Procedia Comput. Sci., vol. 175, pp. 585–590, 2020, doi: 10.1016/j.procs.2020.07.084.

[35] S. Rahman, "Comparative analysis about the challenges and implications of IoT development methodologies," p. 55, 2018.