Advances in Science, Technology and Engineering Systems Journal
Vol. 5, No. 6, 353-362 (2020)
www.astesj.com
Special Issue on Multidisciplinary Innovation in Engineering Science & Technology

ASTES Journal
ISSN: 2415-6698

Blueprint Model: An Agile-Oriented Methodology for Tackling Global Software Development Challenges

Andre Figliuolo da Cruz, Cristiano Pereira Godoy, Lanier Menezes dos Santos, Lucas Frota Marinho, Marco Santarelle Jardim, Elisangela Paiva da Silva, Cícero Augusto Pahins, Paulo Fonseca[*], Felipe Taliar Giuntini

SOL department, Sidia: R & D Institute, Manaus, 69050-020, Brazil

ARTICLE INFO

Article history:
Received: 31 August, 2020
Accepted: 23 October, 2020
Online: 20 November, 2020

Keywords:
Methodology
Agile
Scrum
Kanban
Global Software Development

ABSTRACT

In recent decades, the challenge of managing the development teams spread across different time zones has become increasingly common, raising the importance of the development of Global Software Development (GSD) techniques, in order to tackle its particular problems. This work discuss these issues in the context of Sidia, an R&D institute which implements technological solutions for global companies. The main partner of Sidia is a mobile multinational company located in overseas. The development team must cooperate with the overseas team, even though there are no overlapping working hours between both teams. Besides, the teams have a different set of skills regarding design, quality assurance, and software engineering. In order to address theses problems, we propose the Blueprint model, a Kanban and Scrum based model that supports the development of GSD systems, the allocation of tasks and teams, and the efficient communication. Finally, we discuss the aspects and lessons learned of development of project and deploy of a new model for systems development on a real-word project.

1 Introduction

Many companies adopt software development methodologies with fixed timebox periods for developing incremental deliveries, which is a strategy aligned with agile-based approaches. In these scenarios, Scrum proved to be the *de facto standard* for the most software-oriented industries, showing improved results over *waterfall* methodologies. Despite this, some groups can face difficulties when dealing with the Scrum's ceremonies in the context of Global Software Development (GSD), e.g., estimating effort during the planning sessions or even conducting daily meetings.

In this current work, we extend the definition of the Blueprint model [1], generalizing it to other scenarios and groups by furthering exploring its technical aspects and presenting add-ons to the original proposal. As described in [1] by the authors, Blueprint Model is a lightweight project management that can encourage and facilitate communication between groups in different sites when matched with the suggested group organization. In this particular work, the authors focus on reporting the main challenges when developing a project in collaboration with an external group that does not use agile methodologies, which resulted in a set of barriers to successful integration. To resolve these issues, the Blueprint Model defines its deadlines oriented by the scope of work, which leads to timeboxes of different sizes that consider the effort needed for each activity. Such an approach is different from Scrum, where *sprint* cycles have fixed size throughout the project lifecycle. In this Model, the scope of work, hereafter called *blueprint*, comprises the individual tasks required to reach the target, the necessary information needed to perform these tasks, and what is needed to integrate the work done by every group member. A blueprint defines different group layers, concerning the nature of the work performed (e.g., design group, quality assurance), and assigns the individual tasks to them. Each group layer can be further split into many other groups or even share group members. In this Model, every group must focus on a specific kind of activity since it leads to facilitated project management.

This research was developed on Sidia, an R&D institute located in Brazil, which produces and validates software for an international mobile device manufacturer. The Institute develops innovative software in a kind of areas such as machine learning, software management, and others related to mobile products. Both local and overseas groups deal with the business model integration problems and manage the Sidia's customer's standards for system and management interactions.

In order to meet its customer's demands, Sidia employs Scrum

[*]Corresponding author: Paulo Fonseca, Sidia: R & D Institute, Amazonas, Email: paulo.fonseca@sidia.com

in the context of a global software environment to deal with several model integration problems and manage Sidia and its customers for management interactions. In this scenario, both groups found it difficult to follow Scrum's activities and bureaucracies. In fact that the standard daily meeting system becomes arduous to follow, because, unlike Scrum recommends, groups are not multi-functional. Each group has its own set of skills, including software engineering, design, and quality assurance. Another prominent issue is to apply Scrum in large scale projects, becoming impractical with excessive management tasks and many activities to manage under a distributed scenario.

The remainder of this paper is structured as follows: Sections 2 and 3 summarizes other similar methodologies, relating them to our proposal. Section 4 shows the Blueprint approach and its implications. Section 5 shows a set of results obtained when employing our proposal on a real-world project. In Section 6 we present the conclusions and avenues for future work.

2 Background

In this section, we discuss the frameworks and methods used as references for the development of the Blueprint model, which is base on agile methodologies concepts and rules like those influenced by the values of the *agile manifest*, namely: (i) individuals and interactions over processes and tools, (ii) working software over comprehensive documentation, (ii) customer collaboration over contract negotiation, and (iv) responding to change over following a plan. As already stated, the Blueprint model shares similarities with both Scrum and Kanban.

Scrum is a structural framework created to manage work on complex products and consists of Scrum groups associated with rules, events, roles, and artifacts. Each component owned with purpose, and integration success depends on managing the relationships and interaction between them. Scrum was the primary reference when designing Blueprint.

A fundamental difference between Scrum and Blueprint is that, while the former has been used to develop software, hardware, networks of interacting function, autonomous vehicles, schools, government, and managing the operation of organizations, the latter is focused on software development. We stretch this difference in Section 4. The essence of Scrum is a small group of people while remaining highly flexible and adaptive. To achieve that, it prescribes four formal events for inspection and adaptation: (i) Sprint Planning, (ii) Daily Scrum, (iii) Sprint Review, and (iv) Sprint Retrospective. Appointed events are used in Scrum to create frequency and to reduce the meetings not defined in the regular workflow. In contrast, Blueprint is a model that reduces or eliminates the regularity of those ceremonies, once it can negatively impact some scenarios of GSD. We explore these scenarios in Section 4.4.

To task management, Blueprint employs Kanban, once it is focused on managing commitment and balancing workflow to achieve greater agility. Kanban is a method for defining, managing, and improving services that deliver knowledge work, such as professional services, creative endeavors, and the design of both physical and software products. While designing Blueprint, we make sure that it shares Kanban's same values, namely: transparency, balance, collaboration, customer focus, flow, leadership, understanding, agreement, and respect.

In summary, Blueprint deal with two typical GSD challenges:

Communication Barriers Both Scrum and Kanban define meeting and ceremonious as part of project communication, but in a GSD context, they can be challenging to coordinate, once group members and stakeholders are in different timezones.

Scope Management Although Scrum is favorable to accepting a change of scope, the typical framework implementation implies that a group is committed to the sprint scope once it begins. This scenario can cause undesirable overhead once it requires both backlog and sprint replanning.

3 Related Work

In the last two decades, agile methods increased their popularity and were established as the main software development methodologies. A large number of efforts [2–5] were focused on studying their application under different circumstances and evaluate their advantages and shortcomings. In [4] the authors conducted a systematic review of works that used Scrum guidelines in a global software development (GSD) context. The authors identified the challenges of using Scrum in such context and which are the current strategies to deal with them. The main difficulties are related to: synchronous communication, collaboration difficulties, communication bandwidth, tool support, group size and office space.

The lack of synchronized hours between the group hampers the ability of holding long meetings, such sprint planning or retrospective sessions, which can last up to four hours or more. Some groups [6, 7] handle these issues using strategies such as defining strict timeboxes for meetings' duration and executing preparation tasks in order to shorten the meeting. This approach may help teams to face GSD challenges while partially following Scrum guidelines, however, this is not possible between locations where the overlapping hours are scarce, for example, Brazil and Japan have a 12-hour offset.

Additionally, in many cases, the quality of communication is also a problem, given that distributed projects are composed by group members with diverse cultural and linguistic background, which may lead to group members not completely voicing their opinions, due to fear of being misinterpreted. A number of works [8–10] tackles these issues by proposing additional on-site meetings in multiple project phases: before starting a project, in order to reach a common understanding about the project; multiple exchange visits of group members, during the project, in order to reduce the cultural distance. These studies show that such practices are valuable to both teams as they shorten the communication gap, an important challenge in GSD. Nevertheless, these approaches have a high financial and time cost. As a way to address these issues, other approaches [11, 12] focus on enhance documentation practices in order to bridge the communication gap.

Besides these socio-cultural differences, other inherent aspects of GSD projects also cause Scrum projects to heavily rely on a diverse set of tools in order to exchange information, share group's states and enforce Agile guidelines. In current literature, works as [13–15] used a large number of tools to achieve these goals, including enterprise wikis, shared electronic work space, global issue and bug trackers and backlog management tools, these approaches might be used in order to provide information for distributed teams in a centralized manner. However, these tools might add complexity and bureaucracy to the team operation, negatively affecting its performance.

Another common technique used in large distributed groups is to divide the group into into smaller sub-groups [6, 7, 13], according to different strategies, including: different aspects of the development group(*e.g.* design, back-end, front-end); product features and deliverable; independent modules or subsystems of the main project. This division eases group management due to the smaller gap in communication of a smaller and more focused group. Scrum may also be adapted in different ways to support the groups subdivision: a fully integrated Scrum group, where all members must participate in all meetings, with frequent meeting ensure correct and clear communication between distributed sub-groups; or separated Scrum sub-groups based on location running their own iterations while sharing key points between groups. These subdivisions tackle the management and interdependence problems in GSD because usually team members from the same aspect need to exchange information more frequently than team members from different aspects.

The majority of approaches performs only minor changes in the Scrum framework, by specifying constraints or adding features to original guidelines. Other approaches suggest more fundamental changes to Scrum guidelines or even use other Agile methodologies.

In [16], the authors investigated the impact of using a combination of Scrum and Kanban (Scrumban) on global software development and showed that challenges related to communication, cultural and strategic issues are addressed due to characteristics of Scrumban such as iterative and incremental deliveries, regular feedback ceremonies and limiting the amount of work in progress. However, there are still few challenges related to technical and security issues in GSD that are not addressed or alleviated through the use of Scrumban, which require other methodologies and tools to be mitigated.

In [17], the authors evaluated the use of Kanban on the same environment and found that some Kanban elements, such as the Kanban Board, *Inclusion Criteria*, which guarantees that all items in development deliver value to the customer, and *Reverse Items*, which enables development items to return to a previous state, are valuable to face communication challenges in GSD.

This work proposes a model called Blueprint, which is a lightweight project management that was envisioned specifically aiming global software development scenarios. This combines strengths of Scrum and Kanban, as well as the use additional tools in order to facilitate communication between groups in different sites and provide holistic project view that centralizes information for all team members. In this particular work, the authors focus on reporting the main challenges when developing a project in collaboration with an external group that did not use agile methodologies, which resulted in a set of barriers to successful integration.

4 Blueprint

4.1 Goals

The Blueprint model has been designed to deal with geographically distributed groups, handling timezone issues, which may lead to difficulties in setting up meetings, misunderstanding requirements, delay in receiving or reporting issues. Besides the timezone challenge, this model also focuses on increasing time spent on product development, parallel work, and reduce meetings overhead.

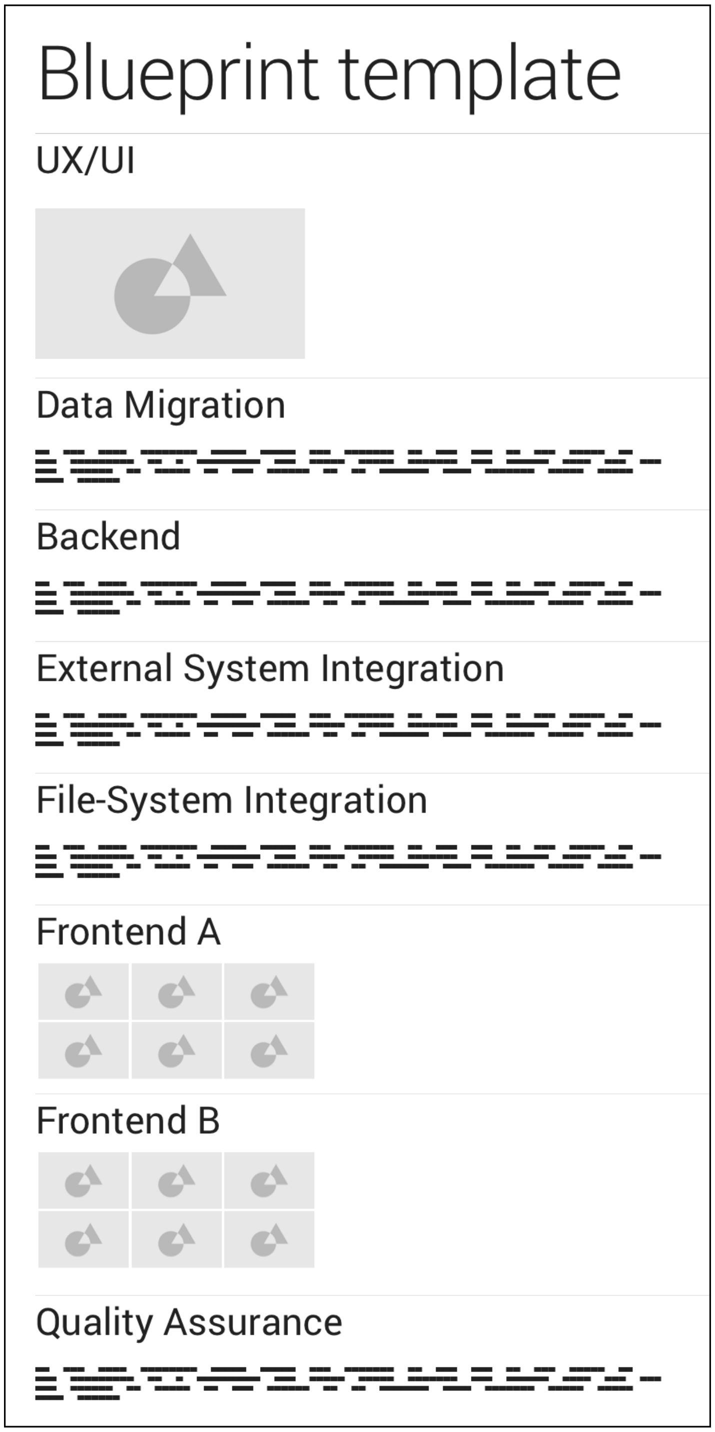


Figure 1: The *Blueprint* Layout, unifying areas *sub-teams* with information on a wiki page.

Therefore, to achieve these objectives, Blueprint seeks to simplify and improve project understanding by using visual-aid tools and helping the group prototype the main project aspects, i.e., UI and technical features. To group communication, these resources are stored in a wiki tool, available to every member so that everyone can be notified of any update.

The Blueprint model employs a hierarchical group organization of sub-groups, once it facilitates the development of unrelated requirements in parallel. Each sub-group is responsible for developing and integrating different aspects of the project into a releasable package. In contrast to other agile methodologies like Scrum, in which groups are considered self-organizing without any prior hierarchical organization [18] other than Testing, Architecture, Operations, and Business Analysis Domains, the Blueprint requires a Leader for each sub-group that is responsible for attending meetings, define requirements, and propose deadlines.

In summary, the primary purpose of Blueprint is to reduce overheads, so the model does not contain predefined which ceremonies must occur neither its regularities, which should be scheduled on demand. Blueprint suggests that each group guides itself by using a set of add-ons, e.g., a wiki tool, improving communication, and information sharing. For the coordinate task, Blueprint suggests Kanban boards, making it easy to share the information across different timezones and provide an accessible project overview.

4.2 Model

In [19], the authors describe an essential aspect of Scrum and Kanban, two popular agile methodologies used in the industry [20], is the constant expansion and improvement of its tools. A Blueprint model is a tool inspired by Scrum and Kanban that manages the project's business entities before it is developed. Blueprint introduces key adaptations to Scrum and Kanban to reduce unwanted bureaucracies and to facilitate global software development. The fundamental concepts of the Blueprint model are:

The Blueprint The concept of a blueprint is very similar to Scrum's story. While both the story and blueprint contain a brief description of the requested features, the latter contains every information needed to design, execute, evaluate, and deliver a set of features, as shown in Figure 3 1. A blueprint should be divided into meaningful sections and contain all related files and discussions to an activity, e.g., the development progress, facilitate the communication between group members and stakeholders, and lower the complexity of both maintenance and monitoring. For efficient scope management, a blueprint can be revised, merged, or divided to reflect the latest requested features and evaluate agreed scope, which should occur at regular intervals. In this document, you can find two very similar terms, the Blueprint Model and the blueprint. The first one, Blueprint Model, is directly related to the model itself. Every time the model is being addressed, it will be called as Blueprint Model, using the first letters for both words in uppercase. The other one is regarding the unity of the work of each blueprint. In this case, the first letter in lowercase refers to every blueprint created for each project, having its characteristics being explained over this document. To implement the blueprint concepts, one could use wiki tools to (i) allow discussions as recorded chats and (ii) keep a history of decisions during development. We further explore the use of wiki tools in Section 4.2.1.

Timebox and Sense of Urgency Timeboxing is a well-known time management technique employed for both project and personal management approaches, once it keeps a constant *workflow* with deliberated deadlines. The use of fixed timeboxes can be challenging, notably in projects with frequent scope changes, since it must be adopted by every group member and stakeholder, imposing difficulties in managing unexpected features requests or modifications. The Blueprint model was designed as a solution for companies and groups struggling to keep with fixed timeboxes, as suggested by agile methodologies like Scrum. In the Blueprint model, every cycle can have an independent deadline determined by the amount of work estimated by the members involved in the development, ranging from days up to weeks. Note that the total time to deliver a set of features is the sum of all activities related to a specific blueprint, similar to a Gantt chart.

Group Organization Contrary to Scrum, in which the group typically uses a flat organization, with members associated with specific roles, the Blueprint model defines a position entitle *group leader*, which is responsible for understanding the solution, define the project schedule, and evaluate deliveries. In the context of a *research and development* project, the Group Leader is also responsible for granting the paper publication. The Blueprint model organizes the group into focused sub-groups, which can be formed by specialists or multidisciplinary members. Sub-groups are advised to define a Sub-Leader to help the group leader in the project management activities, once it eases communication and offers an opportunity for every group member to express its ideas during the discussion of solutions and deadlines. We further explore the Blueprint's group organization in the following sections. Similar to other agile methodologies, the Blueprint model can be adopted by *non-coders*, e.g., testers and designers.

Contrary to the Scrum group division, Blueprint proposes a group hierarchy that can better adapt to software requirements changes during project execution. Blueprint defines its group hierarchy as *sub-groups*. The number of members per *sub-group* is not defined by Blueprint, but as an analogy to Scrum, and as discussed in [19], *sub-groups* should focus on small groups. Also, the number of *sub-groups* may change during the execution of a project.

4.2.1 Add-ons

In the following paragraphs, we present add-ons that can improve Blueprint adoption in different scenarios and groups.

Blueprint Table As explained in Section 4.2, the *blueprint table* is a management tool that should be regularly updated by the group leader to help other members realize the overall status of the project. The *tables* of Blueprint are equivalent to Kanban's boards, but instead of using *status flags* as columns, it uses the concept of *sub-group* ownership. The order of the sub-groups in the table should be defined to reflect the workflow being developed by the group, e.g., design should come first in several common situations, and quality assurance should come last in most of the scenarios.

The list of the condition is used to manage the operation of each stage described in the *blueprint table*, as shown in Table 2.

Contrary to Kanban, in which group members move tasks over the board to indicate the current status of a task, in the blueprint table, the Group Leader is responsible for moving the sub-groups status to consider the *blueprints* in the preferred order of development. The order of development must be arranged between the group, stakeholders, and Group Leader, facilitating project management. The Blueprint model defines the usage of status flags as:

- Every sub-group status should start as PENDING.
- Sub-groups without actions associated with a blueprint, assume the N/A status.
- At the point when the sub-group begins to work in a particular outline, its status should be changed to IN PROGRESS to show that not all conditions and errands are done.
- After a sub-group finishes up all tasks and conditions, the status of the sub-group ought to be set apart as DONE.
- When a sub-group concludes a blueprint, if possible, the sub-group must start working on another blueprint. If not possible, the sub-group must contact the group leader as soon as possible for resource reallocation.
- Status like *SIMILAR* and *BUG FIX* were intended to help general overview.
- SIMILAR ought to be utilized when no effort should be spent by a sub-group due to another outline. At the point when applied accurately, it can diminish modify.

Table 1: *Blueprint* suggestion of content by *subgroups*

Sub-group	Blueprint content
UX/UI Sub-Groups	▪ UX workflow annotation ▪ UI visualizations based on UX definition ▪ Business rules definition ▪ List of assets
Data Migration Sub-group	▪ Database infrastructure servers documentation ▪ Database model related to the Blueprint ▪ Migration architecture documentation ▪ Batch architecture documentation ▪ List of Scripts
Back-end Sub-group	▪ Back-end infrastructure servers documentation ▪ Back-end architecture documentation ▪ Security documentation ▪ Back-end endpoints list
External System Integration Sub-group	▪ External System infrastructure servers documentation ▪ External Integration architecture documentation ▪ External System endpoints list
File-System Integration Sub-group	▪ File-System infrastructure servers documentation ▪ File-System Integration architecture documentation ▪ File-System endpoints list
Front-end A Sub-group	▪ Front-end A infrastructure servers documentation ▪ Front-end A architecture documentation ▪ Security documentation
Front-end B Sub-group	▪ Front-end B infrastructure servers documentation ▪ Front-end B architecture documentation ▪ Security documentation
Quality Assurance Sub-group	▪ Test Cases documentation ▪ Tests Execution report ▪ Know Issue list

- BUG FIX ought to be utilized to show that a blueprint arrived at its last stage and bugs are being fixed.

Every time a new blueprint is created, abandoned or regrouped, it is advised to replan the blueprint table. The group Leader and the group must be aware of the impacts and dependencies of documentation modification during the entire project lifecycle. The same happens when remodeling sub-groups. If any sub-group is created, ceased, or regrouped, the blueprint table should be planned. In summary, a blueprint table is a tool for sub-group management that can be associated with a collection of Kanban boards for handling tasks administration.

Wiki Tools On the Blueprint model, the usage of Wiki tools is essential to organize and aggregate information on different aspects of the project, facilitating the communication between the group and stakeholders. The main points of integration are:

- The web page format makes it easy to maintain and update.
- Every blueprint should be stored as a unique wiki web page.
- The group Leader should organize all pages and ask for the group to keep every part updated.
- Every sub-group must have its own space following the same order on every blueprint page.
- The storage of technical conversations and discussions in a log format contributes to the project's maintenance.
- All related usage of blueprints in the Wiki tools used the company's infrastructure company's infrastructure and its internal Information Security policy to manage it. This means every group member was under the company's management overview and user access control rules.

Table 2: Status to control activities in the *blueprint table*.

Status	Definition
PENDING	Initial status or *sub-group* activities are on hold. It may have several causes. Descriptions should be at the Blueprint related page.
IN PROGRESS	The activities of this *Blueprint* are on development. It may contain dependencies or impediments but still in progress.
DONE	All activities are finished. Have no dependencies.
SIMILAR	When the activities of this *sub-group* are equals to the activities of the *sub-group* above. This eliminates the effort of the related activities.
BUG FIX	Fixing bugs of the *Blueprint*. The *Blueprint* is in its final status.
N/A	Not applied. It occurs when the *sub-group* have no activities for that Blueprint.

- Replanning should be easy to maintain by employing the wiki's versioning system.
- By leveraging wiki tools, groups that require necessary social distancing, home office, or are facing other adverse situations should communicate effectively by (i) integrating a Kanban board with each blueprint and (ii) using the *export* feature to share relevant information with stakeholders.

4.3 Implementing Blueprint on a Real-World Project

To show Blueprint in detail, we will use a real-world project called *The Project*, which characterized a list of *sub-groups* that can be equivalent to many webs, mobile, or embedded software projects. We first planned *The Project* with a number of *sub-groups*, which was thought to be sufficient, but at the end, the list changed to optimize group allocation, as shown in Figure 2. As showed on the Blueprint model, sub-groups can can aggregate multidisciplinary individuals or authorities with a similar profile, for example analyzers situated in Quality Assurance Sub-gathering, architects in Design Sub-gathering, engineers taking a shot at both or solely on Front end Sub-gathering and additionally Back end Sub-gathering. Note that the sub-bunch rundown can be changed by the necessities of the task to streamline bunch portion or build up another prerequisite. In *The Project*, the accompanying circumstances lead to sub-bunch rearrangement:

- To deliver the back-end features on the schedule, the sub-group was divided to help some members to work on two sets of requests.
- The client requested an alternative and entirely different UI solution from Design Sub-group that was approved to be developed, so it was decided to split the Front End Sub-group into groups *A* and *B* to allow each work on a different release.
- A new mobile solution was requested, but there is no developer with such skills in the group. Therefore, it was decided to hire new members and they will work in the entirely new Mobile Sub-group.
- After a schedule modification, the client suggested for developers to work in a new scope of migration data, and due to this, a new Migration Sub-group is created. The group Leader decided that some back-end developers and testers will work on this group part-time.

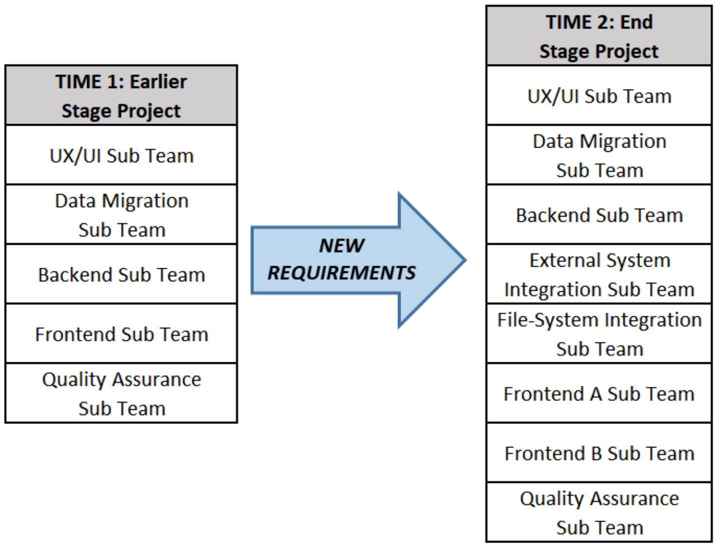


Figure 2: *Sub-groups* preparations during beginning (*time 1*) and ending (*time 2*) of *The Project* execution.

Blueprint consents to the thought that the quantity of individuals doesn't speak to quality or profitability. In the Blueprint model, each *sub-group* has a pioneer characterized as a part of mindfully, and not really work execution. Each *sub-bunch leader* is liable for the exercises and consequences of its gathering. He is additionally answerable for incorporating the errands done by the gathering into the undertaking, planning this cycle with different pioneers. All *sub-bunch leaders* react to the *group leader*, as delineated in Figure 3. This association follows the meaning of how improvement ought to be executed by the Blueprint model to permit and encourage *sub-group* communication.

After the meaning of *sub-bunch leaders*, the following stage is to execute Blueprint select and the board. This stage is a variation and conglomeration of two Scrum the board undertakings and ceremonies, in particular, *Product Backlog* and *User Stories*. All individuals are locked in to effectively take an interest in this stage during venture improvement, as it is the center of the model. As depicted in [12], the duty to characterize and keep up the Scrum's *Sprint Backlog* is in the possession of the job *Product Owner*. It is normal for the *Product Backlog* to require consistent correction and reworking. These assignments may contain business rules in the *User Stories'* depiction. A few organizations have utilized extra documentation to keep engineering definition, model plan, and framework examination refreshed by the *Product Owner*. Uniquely in contrast to Scrum, *blueprints* are business substances that are relied upon to be persistently changed, kept up, and assessed by all individuals from the task. Pioneers must have the duty to change the *blueprints*, however this assignment isn't limited to them.

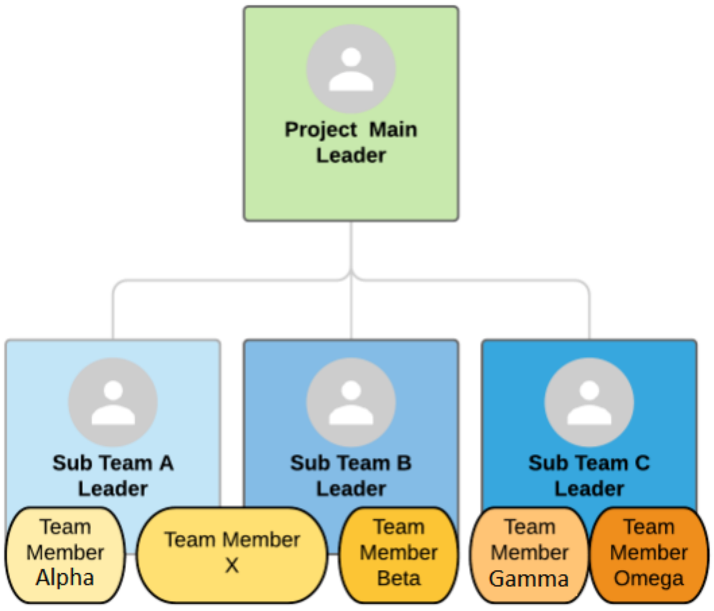


Figure 3: *Sub-group Leaders* and their relationship with the *main group Leader*.

As presented in Section 4.1, the Blueprint model was intended to help bunches in various destinations and to function admirably with distant *sub-groups*. It energizes correspondence among designers and customers by incorporating the conversation in a wiki-based website page. Each *blueprint* ought to be enlisted utilizing a wiki instrument, and its substance development relies upon each *sub-group's* work. At *The Project*, *blueprints* are wiki pages containing all substance vital for the advancement of a module or an aspect of the undertaking. Table 2 sums up the substance of each *sub-group* obligations that was characterized in *The Project*.

Blueprints must contain all *sub-groups* related materials, as tables, diagrams or file that aggregates information to development. *Blueprints* centralizes information and communication between *sub-groups* as well as with the *group leader* and the client. Wiki tools typically allow for managing discussions between users as a recorded chat. Blueprint takes advantage of this feature to keep the history of decisions made by the group during project execution. Figure 1 illustrates a *blueprint* template that contains fields related to each *sub-group* define in *The Project*.

Blueprint maintenance is done by keeping each blueprint activity from all sub-groups up to date, which helps reflect on the project status. The usage of blueprints encourages deadlines with a focus on the agreement between sub-groups, group leaders, and stakeholders. Each sub-group deadline is a composition of all tasks planned at the blueprint, which can be replanned without impacting other blueprints. Therefore, the Blueprint model enables groups and stakeholders to control and refactor activities by proposing specific deadlines for each sub-group without re-planning activities defined.

The understanding about the hour of every movement is legitimately identified with the development of the gathering and its self-administration. After the meaning of each *blueprint* content and the principle cutoff time of *The Project*, some gathering individuals recognized the requirement for micromanagement exercises. To oblige that request, the underlying adaptation of the *blueprint table* was moved up to characterize reasonable cutoff times for all *sub-groups*. Figure 5 embodies the second form of the *The Project*'s *blueprint table*.

Blueprint encourages deadlines with a focus on *sub-group* assignments that are characterized in in agreement with the *group leader* and the client. Each *sub-group* deadline is a piece of all assignments arranged at the *blueprint*. During the development of *The Project*, bunch individuals started to demand rethinking of certain activities, and it started to impact on previously defined *blueprints*. The Blueprint empowers to gatherings and customers to control and refactor exercises by proposing specific deadlines for each *sub-group* without previously characterized.

The Blueprint model characterizes a*blueprint table* as a administration tool hat ought to be utilized to continually show the general status of an undertaking to all gathering individuals and customers. Figures 4, 5 and 6 show the *blueprint tables* during execution of *The Project*. Toward the start of *The Project*, an underlying form of the *blueprint table* was characterized by the gathering with the assistance of the customer, as observed in Figure 4. This underlying arrangement permitted all partners to concur with the interests of the task and with the substance of each *blueprint*. During the development of *The Project*, members of the *sub-groups* identified that the *SIMILAR* status helped to reduce software rework.

After different steady modifications, *The Project* characterized its last *blueprint table* as appeared in Figure 6. Note that, in examination with Figure 5, new *sub-groups* were made to oblige already impromptu prerequisites of the customer. The creation or evacuation of *sub-groups* can influence different exercises and should be all around oversaw and talked about between *sub-bunch leaders*, however never debilitated. For example, at the underlying rendition of the *blueprint table*, just one front-end gathering, called *Front-end Sub-group* was liable for all front-end exercises. At that point, when an additional necessity rose up out of the customer, another front-end *sub-group* called *Front-end B Sub-group* should have been made to build up an alternate arrangement of arrangements. The underlying front-end *sub-group* was needed to survey its exercises to help the redistribution of errands of the reestablished *blueprint table*. A portion of the gathering individuals need to chip away at bunches *A* and *B* to help extra exercises. *Front-end bunch B Sub-group* was situated at the abroad site and worked distantly with the gathering situated at Sidia. The last rundown of *blueprints* are represented in Table 3.

Table 3: List of Blueprints used in *The Project*

Blueprint Name	Example of work by blueprint
Authentication	Login page. Back-end authentication service. Front-end authentication error page.
CRUD A, B, C and D	Create, Read, Update and Delete business entity A, B, C and D. Back-end service for each. Front-end pages for each.
Extended Specification A and B	New requirement with several business workflow. Requested during the execution of the project. Replan needed.
Report A, B and C	Report generator page. Back-end service for report business model.
External Requests A and B	Unintended external integration. Requested during the execution of the project. Replan needed.
Mobile Integration and Mobile Interface	Unintended mobile version. Requested during the execution of the project. Replan needed.
Customer Service, User Overview and About	Low priority administrator pages. Gained priority due to requested changes.

The Blueprint Model as some other philosophy has its reception challenges, for instance:

- Communication hazards The Blueprint model has a great dependency on documentation process. As proposed, it is very dependent of Wiki documents in order to keep good communication between sites. This dependency combined with possible many changes requests, generates a constant necessity of Wiki documents updates. It can turn out very difficult for the group to keep the good communication under many changes requests.

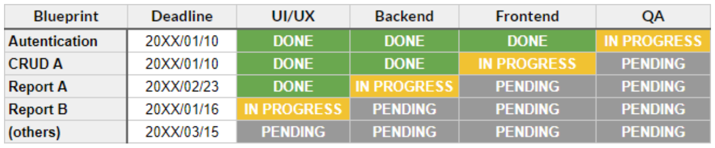
- Predictability hazards If not applied as its standards by the entire group, the Blueprint model can show up as a harmful tool when it comes to deliveries and projects goals. The presented model requires management skills from each area of the group. Once the required skills are not present in the group members, the entire process can be compromised, which would impact on the completion and good quality of deliveries.

- Performance hazards Concerning software development, all sub-groups that work together have dependency of information from one another. On Blueprint model, as each sub-group is required to maintain its own Wiki page updated, the lack of update of any sub-group can generate delays, compromise group work and even entirely block group members of working.

4.4 Use Cases

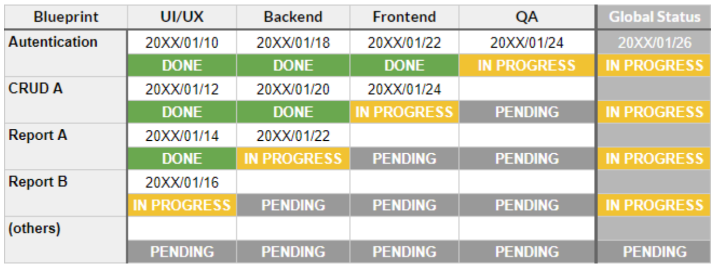
In the following paragraphs, we present some use cases in that the Blueprint model proves successful.

Global Software Development The Blueprint model was designed to support remote groups and sub-groups in different locations, since its hierarchical group organization of a Project Leader and its respective Sub-groups facilitates daily communication, in such a way that this interaction becomes more fluid and objective for adopting a particular scope for each sub-group. In our case, we deal with a customer in a country in which the timezone is at least 13 hours ahead, making real-time communication via call or chat unfeasible. In this scenario, we use the 'wiki' to keep track of what each member of the group is planning for the next few days, what they are doing, and what was done in such a way that everyone has an overview of the project's progress, being able to follow all deadlines and features that have already been developed and those to come.



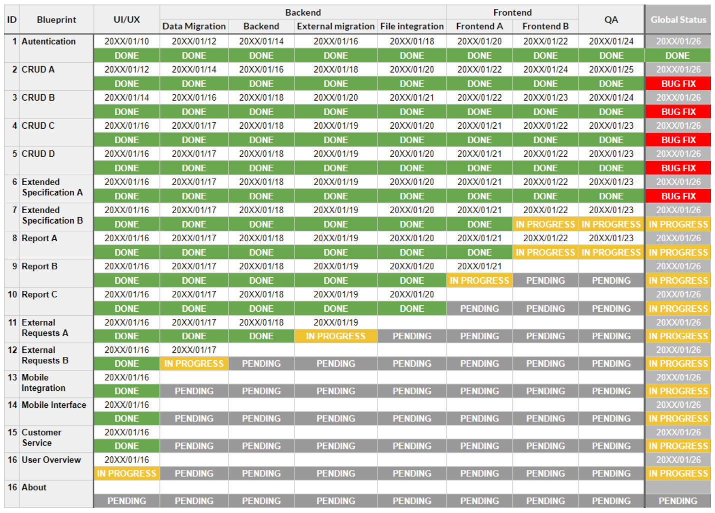
Blueprint	Deadline	UI/UX	Backend	Frontend	QA
Autentication	20XX/01/10	DONE	DONE	DONE	IN PROGRESS
CRUD A	20XX/01/10	DONE	DONE	IN PROGRESS	PENDING
Report A	20XX/02/23	DONE	IN PROGRESS	PENDING	PENDING
Report B	20XX/01/16	IN PROGRESS	PENDING	PENDING	PENDING
(others)	20XX/03/15	PENDING	PENDING	PENDING	PENDING

Figure 4: Starting version of the *blueprints table*. Deadlines are characterized for *blueprints* in *The Project*.



Blueprint	UI/UX	Backend	Frontend	QA	Global Status
Autentication	20XX/01/10	20XX/01/18	20XX/01/22	20XX/01/24	20XX/01/26
	DONE	DONE	DONE	IN PROGRESS	IN PROGRESS
CRUD A	20XX/01/12	20XX/01/20	20XX/01/24		
	DONE	DONE	IN PROGRESS	PENDING	IN PROGRESS
Report A	20XX/01/14	20XX/01/22			
	DONE	IN PROGRESS	PENDING	PENDING	IN PROGRESS
Report B	20XX/01/16				
	IN PROGRESS	PENDING	PENDING	PENDING	IN PROGRESS
(others)					
	PENDING	PENDING	PENDING	PENDING	PENDING

Figure 5: Second version of the *blueprint table*. The deadlines were characterized by *sub-groups*.



ID	Blueprint	UI/UX	Backend				Frontend		QA	Global Status
			Data Migration	Backend	External migration	File integration	Frontend A	Frontend B		
1	Autentication	20XX/01/10	20XX/01/12	20XX/01/14	20XX/01/16	20XX/01/18	20XX/01/20	20XX/01/22	20XX/01/24	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE
2	CRUD A	20XX/01/12	20XX/01/14	20XX/01/16	20XX/01/18	20XX/01/20	20XX/01/22	20XX/01/24	20XX/01/25	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	BUG FIX
3	CRUD B	20XX/01/14	20XX/01/16	20XX/01/18	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/24	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	BUG FIX
4	CRUD C	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	BUG FIX
5	CRUD D	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	BUG FIX
6	Extended Specification A	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	DONE	DONE	BUG FIX
7	Extended Specification B	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	IN PROGRESS	IN PROGRESS	IN PROGRESS
8	Report A	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21	20XX/01/22	20XX/01/23	20XX/01/26
		DONE	DONE	DONE	DONE	DONE	DONE	IN PROGRESS	IN PROGRESS	IN PROGRESS
9	Report B	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20	20XX/01/21			20XX/01/26
		DONE	DONE	DONE	DONE	DONE	IN PROGRESS	PENDING	PENDING	IN PROGRESS
10	Report C	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19	20XX/01/20				20XX/01/26
		DONE	DONE	DONE	DONE	DONE	PENDING	PENDING	PENDING	IN PROGRESS
11	External Requests A	20XX/01/16	20XX/01/17	20XX/01/18	20XX/01/19					20XX/01/26
		DONE	DONE	DONE	IN PROGRESS	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
12	External Requests B	20XX/01/16	20XX/01/17							20XX/01/26
		DONE	IN PROGRESS	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
13	Mobile Integration	20XX/01/16								20XX/01/26
		DONE	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
14	Mobile Interface	20XX/01/16								20XX/01/26
		DONE	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
15	Customer Service	20XX/01/16								20XX/01/26
		DONE	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
16	User Overview	20XX/01/16								20XX/01/26
		IN PROGRESS	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	IN PROGRESS
16	About									
		PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING	PENDING

Figure 6: Last version of the *blueprint table*. New *sub-groups* and *blueprints* are remembered for correlation with past tables.

Remote Work In the face of COVID-19, several development groups and companies in different industries were forced to adopt remote work to continue their work. In this scenario, Blueprint proved to be an adequate strategy to accommodate the remote work regime, regarding the difficulty in communication between groups because from the moment employees are working from home, the practicality and speed of transmitting information are both affected. Consequently, adopting a standard of documenting the information in a centralized form and bringing the division of sub-groups, makes the dissemination of internal information through the wiki page more effective by leveraging a reliable consumption source that is validated by the client.

Scope Change Control The Blueprint also helps to manage frequently *scope changes* since unlike Scrum which suggests another iteration of a set of ceremonies when facing modified requests, such as *Project Vision*, *User Stories*, *Planning*, and others, the Blueprint offers lightweight project management that avoids these ceremonies that can cause rework and further prolong delivery forecasts. To improve on that, Blueprint proposes the idea that both the developer and client will leverage the 'wiki' as a guide for (i) decision making, (ii) discussions about new releases, and (iii) requirements alignment. By documenting information in a centralized repository, the stakeholders are able to easily monitor the requested scope changes, helping to avoid noise or loss of information from typical verbal communication.

5 Implementation Findings

The design of Blueprint was mainly oriented by the GSD challenges faced during the collaboration of remote teams on developing *The Project*, which evolved from 10 members using Scrum to a successful Blueprint case more than 20 members. The increased number of members resulted from the stakeholders' higher engagement and contribution to the process and project. Once the process started running, it allowed the team to accept more work scope, which might be understood as an increase in productivity and led to a natural team increase. Through the advancement of *The Project*, different gradual modifications were made to the Blueprint Model to energize its appropriation. This cycle included both neighborhood and far off groups to facilitate the work process, encourage worldwide programming designing, and improve appropriated correspondence. Following the summarizing of *The Project*, we notice various situations to improve the Blueprint model execution to ensuing tasks.

We found that an essential aspect of a successful Blueprint implementation is to focus on the member's sense of integration and empowerment. Regarding *The Project*, most of the individuals concurred they turned out to be more critical and made more proposals to the undertaking the executives with the appropriation of the Blueprint Model. Every part had the option to share its answers and thoughts to other people, and due to Blueprint central focuses, significant correspondence issues were survived. The primary advantage of zeroing in on the part's feeling of reconciliation and strengthening is that the group felt inspired for monitoring Blueprint's curios, prompting improved undertaking the executives.

Another significant part of a successful usage was the group's adherence to Blueprint's rehearses. Regularly, less connected with *sub-group leaders* portrayed exercises and progress ineffectively, which can be credited to the difficulties while embracing another administration model. By the by, the individuals saw a connection between (I) the time frame pages changes, (ii) nature of these pages and (iii) commitment of the sub-pioneers. We notice a connection between diminishing the nature of Blueprint pages to diminishing code quality and last on bugs appearing.

The improvement of the communication between team and stakeholders is a crucial objective of the Model, so it is crucial to make more accessible the verification and reporting of information through a *wiki tool*. The successful implementation of the Blueprint Model enables identifying disagreements and conflicts in the Wiki and not necessarily during meetings, which enabled the resolution of these problems in private. Meeting events are not often and gathers a few members, which prevents long discussions of non-related subjects from reducing the execution time and reaching its goal as quickly as possible while the rest of the team keeps working on project tasks. After adopting the BLuePrint Model, the team had performance gains related to the planned x performed activities, related to clear and accurate communication, thus ensuring a clear understanding of the changes. The fact that did not happen using the Scrum model, due to the difficulty of managing constant changes within the sprint and the difficulty of formal meetings due to the time zone. Another improvement that is possible to observe is the quality of deliveries. In the BluePrint model, team members can maintain their specialties without the need to be multidisciplinary, and brings a significant gain in side effects, especially when it came to scope changes. Still linked to the specialty of the team members, it is also possible to see the result in the motivation of the team; it means, to work with the specialty of each one, this brings motivation and engagement in the day-to-day work, and consequently, this is reflected in product quality delivered.

Differently from other agile methodologies like Scrum, the Blueprint Model exempts ceremonies and daily meetings that need the entire team to happen, making the Model more flexible. This characteristic praises the Model because it fits well with the current pandemic scenario where the new common sense of IT company members working remotely and need to keep the flow of work to meet delivery deadlines. Some Blueprint Model positive points:

- No team ceremony: The team adopted the Blueprint Model since the beginning of our development, and we kept it when everyone on the team was doing home office. We managed to keep all deadlines using only the Wiki as a reference for the division of activities.
- Sub team division: This sub-team division was of great value for the atypical home office scenario. The team needed to maintain a very similar communication level as a live-work environment, which made the team feel more comfortable maintaining contact with companions.
- Model suited to a project scope that frequently changes: Scope change is passive to happen very commonly in any software development project. The team validated the model adherence to the common practice since every project was constantly changed. Due to the very centralized information on the Wiki, there were small relevant impacts, all highly negotiable.

6 Conclusion

In this work, we presented the extended definition of the Blueprint model, a management methodology designed to accommodate the best techniques and habits of both Scrum and Kanban to facilitate (i) global software development, (ii) groups and task allocations, and (iii) effective members communication. The straight communication between groups and the high visibility of the overall status of the projects are the main highlights of the methodology, which lead to a better scope change management with low bureaucracy and an accessible understanding of the entire development. To validate the model, we discussed its adoption on a real-world project of a large mobile-related software development company in the context of GSD. The strategic use of Blueprint's add-ons enables us to coordinate, design, develop, and test a complex set of features on a highly unstable scope scenario. Blueprint usage of UI and UX for the documentation of analysis, design, and development phases in conjunction with the lightweight cycle management tool helped to facilitate the project management, once it allows sharing responsibility between the *group Leader*, *Sub-group Leaders*, and other members in documenting and reviewing the progress of the project. More than that, the Blueprint model was improved during the execution of the project through a collaborative effort of groups located in different countries in order to improve its main benefits and to lower any adoption barriers. As future work, we plan to extend the Blueprint's add-ons to integrate other easy-to-use and well-known tools of project and task management, as well as to improve the model's adherence to other types of non-code projects.

Conflict of Interest The authors declare no conflict of interest.

Acknowledgment We thank MSc. Ilsun Hong, Taehee Kim, Rafael S. Zerbini, Marcos R. G. B. Junior and Alison L. Garantizado for their valuable help during the development of this framework. This work was partially supported by Samsung Eletrônica da Amazônia Ltda., under the auspice of the informatics law no 8.387/91.

References

[1] C. P. Godoy, L. M. Santos, A. F. Cruz, R. S. Zerbini, E. P. Silva, C. A. L. Pahins, "Blueprint Model: A New Approach to Scrum Agile Methodology," in Proceedings of the 14th International Conference on Global Software Engineering, ICGSE '19, 85–89, IEEE Press, 2019, doi:10.1109/ICGSE.2019.00014.

[2] I. Zada, S. Shahzad, "Issues and implications of scrum on global software development," Bahria University Journal of Information & Communication Technologies (BUJICT), **8**(1), 2015.

[3] L. Silva, C. Santana, F. Rocha, M. Paschoalino, G. Falconieri, L. Ribeiro, R. Medeiros, S. Soares, C. Gusmão, "Applying XP to an Agile–Inexperienced Software Development Team," in International Conference on Agile Processes and Extreme Programming in Software Engineering, 114–126, Springer, 2008, doi:10.1007/978-3-540-68255-4_12.

[4] E. Hossain, M. A. Babar, H.-y. Paik, "Using scrum in global software development: a systematic literature review," in 2009 Fourth IEEE International Conference on Global Software Engineering, 175–184, IEEE, 2009, doi:10.1109/ICGSE.2009.25.

[5] J. López-Martínez, R. Juárez-Ramírez, C. Huertas, S. Jiménez, C. Guerra-García, "Problems in the adoption of agile-scrum methodologies: A systematic literature review," in 2016 4th international conference in software engineering research and innovation (conisoft), 141–148, IEEE, 2016, doi: 10.1109/CONISOFT.2016.30.

[6] M. Paasivaara, S. Durasiewicz, C. Lassenius, "Distributed agile development: Using scrum in a large project," in 2008 IEEE International Conference on Global Software Engineering, 87–95, IEEE, 2008, doi:10.1109/ICGSE.2008.38.

[7] J. Sutherland, A. Viktorov, J. Blount, N. Puntikov, "Distributed scrum: Agile project management with outsourced development teams," in 2007 40th annual Hawaii international conference on system sciences (HICSS'07), 274a–274a, IEEE, 2007, doi:10.1109/HICSS.2007.180.

[8] M. Summers, "Insights into an Agile adventure with offshore partners," in Agile 2008 Conference, 333–338, IEEE, 2008, doi:10.1109/Agile.2008.37.

[9] E. Therrien, "Overcoming the challenges of building a distributed agile organization," in Agile 2008 Conference, 368–372, IEEE, 2008, doi:10.1109/Agile.2008.9.

[10] M. Cottmeyer, "The good and bad of Agile offshore development," in Agile 2008 Conference, 362–367, IEEE, 2008, doi:10.1109/Agile.2008.18.

[11] M. Vax, S. Michaud, "Distributed Agile: Growing a practice together," in Agile 2008 Conference, 310–314, IEEE, 2008, doi:10.1109/Agile.2008.44.

[12] M. Hron, N. Obwegeser, "Scrum in practice: an overview of Scrum adaptations," in Proceedings of the 51st Hawaii International Conference on System Sciences, 2018, doi:10.24251/HICSS.2018.679.

[13] W. Williams, M. Stout, "Colossal, scattered, and chaotic (planning with a large distributed team)," in Agile 2008 Conference, 356–361, IEEE, 2008, doi:10.1109/Agile.2008.25.

[14] J. Cho, "Distributed Scrum for large-scale and mission-critical projects," AMCIS 2007 Proceedings, 235, 2007.

[15] J. Sutherland, G. Schoonheim, M. Rijk, "Fully distributed scrum: Replicating local productivity and quality with offshore teams," in 2009 42nd Hawaii International Conference on System Sciences, 1–8, IEEE, 2009, doi: 10.1109/HICSS.2009.225.

[16] A. Banijamali, M. O. Ahmad, J. Similä, M. Oivo, K. Liukkunen, et al., "Empirical investigation of scrumban in global software development," in International Conference on Model-Driven Engineering and Software Development, 229–248, Springer, 2016, doi:10.1007/978-3-319-66302-9_12.

[17] M. Tanner, M. Dauane, "The Use Of Kanban To Alleviate Collaboration And Communication Challenges Of Global Software Development," Issues in Informing Science & Information Technology, **14**, 2017, doi:10.28945/3716.

[18] S. V. Spiegler, C. Heinecke, S. Wagner, "Leadership gap in agile teams: how teams and scrum masters mature," in International Conference on Agile Software Development, 37–52, Springer, Cham, 2019, doi:10.1007/978-3-030-19034-7.

[19] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," Robotics and Computer-Integrated Manufacturing, **43**, 59–67, 2017, doi: 10.1016/j.rcim.2015.12.001.

[20] M. Düchting, D. Zimmermann, K. Nebe, "Incorporating user centered requirement engineering into agile software development," in International Conference on Human-Computer Interaction, 58–67, Springer, 2007, doi: 10.1007/978-3-540-73105-4_7.