HYBRID PRACTICES IN GLOBAL SOFTWARE DEVELOPMENT: A SYSTEMATIC LITERATURE REVIEW

Rafael Camara, Iury Monte, Annelyelthon Alves and Marcelo Marinho

Department of Computer Science, Federal Rural University of Pernambuco,
Recife, Pernambuco, Brazil

ABSTRACT

Although agile methods in their purest way fit several companies, it has been a challenge to perform them in environments with distributed teams developing large software applications. Contractual items, for projects under development for external organizations, introduce additional complexities for pure agile-based approaches. The majority of global teams and companies use hybrid development practices that combine different development methods and frameworks. This research provides results from an empirical field study on how the hybrids practices are adopted in Global Software Development (GSD) projects. A systematic literature review was conducted to capture the status of combining agile with plan-driven in GSD projects. The results were limited to peer-reviewed conference papers or journal articles, published between 2001 and 2020. The present study selected 37 papers from five different bibliographic databases. In the end, 16 practices were summarized and described as hybrid by GSD projects. Based on the findings of this study, the authors can conclude that the contribution of this study is not only limited to identifying how hybrid development practices are applied in GSD but also allowing that practitioners can have a basis for adapting their development methods.

KEYWORDS

Agile software development, Hybrid development method, Systematic literature review, Global software development.

1. INTRODUCTION

Many companies foster global software development (GSD) to benefit from cheaper, faster, and better software development [1, 2]. Agile adoption in the industry is growing year by year. However, agile methods tend to rely on informal processes and regular face-to-face communication to facilitate coordination, whereas distributed software development relies on formal mechanisms [3].

GSD requires some tailoring of agile methods that were not intended for such settings [4]. While agile methods have moved well beyond this, with frequent use of agile methods in projects within globally distributed teams, many challenges remain, such as those on cultural differences, breakdowns in communication, and optimum practices for distributing development work across sites [5].

There is no denying that agile methods have become an important asset in the process portfolios of many companies [6]. However, agile methods are not implemented as prescribed by the authors or standards in 2001 [7]. Today, companies often use highly individualized processes to execute projects, generally integrating agile methods in their processes [4].

DOI: 10.5121/ijsea.2022.13101

Recent research provides evidence on the use of hybrid methods in industry, i.e., methods that are combinations of multiple development methods and practices [4, 8–10]. Kuhrmann et. al. [9] proposed the following definition: 'A hybrid software development approach is any combination of agile and traditional (plan-driven or rich) approaches that as organizational unit adopts and customizes to its own context needs (e.g., application domain, culture, process, project, organizational structure, techniques, technologies, etc.)'

Thus, we assume that hybrid practices are defined as is any combination of agile and traditional practices customizes to their own context needs. That is, a practice can be nominated as agile, but it is not adopted in a purely agile way and vice versa. Hybrid practices are showing to be suitable for GSD projects since they require a higher level of coordination and collaboration among teams than regular co-located projects [2]. The combination of traditional and agile practices has shown the capability to lead distributed teams to achieve the success of their projects [11, 12].

While the research literature contains several surveys and some case studies on hybrid approaches adoption [13, 9, 10, 8], a systematic overview, and synthesis of how the hybrids practices are being adopted in GSD projects is lacking. In this paper, we start filling in this gap by presenting a systematic literature review of hybrid practices in the GSD context.

This paper is organized as follows: in Section 2, we introduce the background to the problem and define our research questions. Section 3 describes the method we apply. Sections 4 and 5, present the results, their implications and limitations, respectively. Finally, in section 6 presents conclusions and future research directions.

2. BACKGROUND

2.1. Global Software Development

Global software development (GSD) designates the software development approach where the software is developed across different boundaries, and possibly with teams from different time zones [14, 15]. GSD is commonly being used by companies around the world that aim to utilize global resources and 24/7 software development to achieve higher levels of efficiency, reduce costs, and access global talent [16]. Distributed software development differs of Global software development due to the geographical distribution of the teams involved. Teams distributed across the national boundaries of a company are categorized as GSD, although when a company has distributed teams inside the same country its characterized as distributed software development (DSD) [5]. GSD differs of DSD on its challenges, global teams most of the time deal with cultural, linguistic, and time zone differences instead of DSD teams that are mostly composed by members from the same country [2].

2.2. Hybrid Development Approaches

Hybrid software development approaches are categorized by the combination of agile together with traditional approaches, such as plan-driven [17], waterfall [18, 19], or RUP [20, 12]. The hybrid approaches are commonly used by teams that want to combine the best practices from them in their development process. The use of hybrid development approaches is well received in distributed software projects since they can be benefited from agile practices that allow the concurrent development of functionalities among teams and releases on-demand [20]. Meanwhile, the distributed teams can face different time zones that make impossible the presence of all members at meetings, although they can get also benefited from traditional practices, such as documentation, that was seen as a common practice to make information available to everyone in the project [12] and avoid overlap in work hours.

Hybrid practices can be a way to deal with some challenges in GSD projects since it combines agile and plan-driven techniques to handle coordination, organizational and collaboration issues on GSD teams [11, 12].

2.3. Related Works

There are several articles related to the use of hybrid approaches in the software development process. However, the applicability of such hybrid approaches to GSD settings are still needed.

Kuhrmann et al. [9] conducted a study that present results from the HELENA (Hybrid Software and System Development Approaches) survey, in order to ad- dresses the development of software and systems that face multiple challenges in rapidly changing markets. To address these challenges, companies seek to adopt specific development approaches combining well-structured methods and flexible agile practices. They show which approaches are used in practice and how they are combined during the development. The results found from 69 participants show a variety of development in the approaches used. A pattern was found where the traditional model serves as a structure in which several agile practices are connected. It is also pointed out that hybrid software development approaches are independent of the size of the company and external triggers.

Paolo Tell et al. [21] provides a basis for understanding what structures, meth- ods, and practices are used in practice to execute hybrid methods, in addition to providing evidence-based characterization for these methods. The use of hybrid development methods did not depend on the size of the company. There are eight basic methods (Classic Waterfall, Scrum, iterative development, Extreme program- ming, Kanban, DevOps, Feature Driven Development, Lean Software Development) that are repeatedly combined to form hybrid development methods. An analysis of 36 practices and their relationship to the methods and method combinations was carried out, and few practices were found with at least 85% agreement among practitioners. Finally, through the analysis procedures, the authors could define a statistical construction procedure to describe hybrid methods.

Marinho et al. [4] shows that hybrid methods is common in GSD projects. The authors showed that 72% of globally distributed projects implement hybrid approaches. Besides that, [4] indicate that activities are implemented in a balanced way between agile and traditional methods. However, management activities such as Risk Management and Quality Management are more traditionally oriented, whereas development activities, such as Implementation/Coding and Integration/Testing, tend to be conducted in a more agile manner.

Klu¨nder et al. [10] shows that the main reasons for adopting hybrid methods are to increase productivity, improve product quality, have better planning and project estimation, in addition to increasing the frequency of deliveries to the customer. As evidenced by Marinho et al. [4] shows that hybrid methods improve productivity; planning and estimation; external product quality; frequency of delivery to the customer; adaptability and flexibility of the process to react to change and Time to market.

3. RESEARCH METHOD

Systematic literature reviews are a means of identifying, evaluating, and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. They are appropriate for summarizing existing research, for identifying gaps in the existing literature, as well as for providing background for positioning new research [22].

In this paper, we present the results of a systematic literature review on the topic of hybrid approaches. We sought to answer the following research question: "What practices are used by GSD projects that employ such a hybrid approach?". A practice is a particular technique for developing software, such as Test Driven Development or pair programming, that may be part of many different methods. In this context, we define a method as an approach to managing a software project that uses a set of software development practices in a certain way; examples of methods are Scrum, Waterfall, and Feature Driven Development [4].

We used the following Boolean search string to ensure that we captured a wide variety of papers: ((agile OR scrum OR "extreme programming" OR "plan-driven" OR 'plan driven" OR hybrid OR "hybrid software development" OR "lean develop- ment" OR "lean software development") AND ("global software engineering" OR "global software development" OR "distributed software engineering" OR "distributed software development" OR "GSE" OR "GSD" OR "distributed team" OR "global team" OR "dispersed team" OR "spread team" OR "virtual team" OR "offshore" OR "outsource"))

We used this string to search the metadata relating to journals and conference proceedings in IEEEXplore, ACM Digital Library, SpringerLink, Scopus, and Wiley bibliographic databases.

3.1. Document Selection

The set of potential primary studies identified by the database search was refined in two steps, first by filtering based on titles and abstracts and finally based on the full text. Then, the inclusion and exclusion criteria of the original study were applied.

Inclusion/Exclusion criteria The following criteria guided the selection of papers that helped us address the research questions.

We included : (i) complete, peer-reviewed, published articles; (ii) papers directly related to the research questions; (iii) papers that address agile and plan-driven practices in GSD; and (iv) the study is available via the university library services accessible to the authors during the time of the research.

We excluded : (i) texts not published in English; (ii) technical content, without proven scientific relevance, e.g: editorials, tutorials, key-note speech, white papers, thesis, dissertations, technical reports, books; (iii) short papers (¡=4 pages); and (iv) articles that are not clearly related to the research questions.

The search produced 1949 references. Each potentially relevant paper was analyzed by all researchers and revised by a fourth researcher.

Before accepting a paper into each phase of the review, we checked to ensure that there was no replication. For example, if a given study was published in two different journals with a different order of primary authors, only one study would be included in the review; this would usually be the most comprehensive or recent study. Besides, we checked to ensure that there was no duplication. For example, in the same paper listed in more than one database, only one study would be included in the review.

With these criteria, we identified 89 duplicate articles and no replicates. After excluding duplicate results from the dataset, we identified articles for inclusion in the selection. Of these articles, 37 were finally passed on to the data extraction and data synthesis phase (See Table 1).

3.2. Data Extraction

Data extraction refers to the recording of all relevant information from the studies required to answer the RQ [23]. In order, to have a structured data extraction process and to ease the management of the extracted data, we decided to use the strategy of categorizing studies into facets, as suggested by [24] and [25].

Table 1. Papers by engine.

Engine	Automated Search	Study Selection
ACM	139	7
Scopus	497	9
Wiley	331	0
Springer	203	11
IEEE	779	10
Duplicates	89	0
Total	**1949**	**37**

To record the extracted data from each study a datasheet was used. Quotes from each study that answers the research question was recorded on separate results form. We also synthesized the data by identifying themes emanating from the findings reported in each accepted paper. To reduce the bias of the data extraction results, three researchers performed the data extraction independently. Before the formal data extraction process, all researchers discussed the definitions of the data items to be extracted to ensure that all researchers had a common understanding. After we completed the data extraction, a discussion was held to resolve conflicts for reaching a consensus on the data extraction results.

4. RESULTS

4.1. Overview of the Selected Studies

From the 37 studies selected, most of them were classified as qualitative studies, with 24 papers. 11 studies were classified as mixed which combined qualitative and quantitative methods. We classified all included papers into one of the research type facets derived from [25]. There were no studies that fit as opinion papers. The most common research facets were evaluation, with 15 papers [20, 26–32, 15, 33, 19, 34–37], followed by experience with 9 [38, 14, 39, 11, 40–44], then philosophical with 8 [45, 46, 16, 17, 47–50], and finally 5 from solution facet [18, 51, 52, 12, 53].

Also, all the reviewed studies were classified through contribution type facets derived from [24]. None of the SLR papers could be classified as a theory or a tool. The most present contributions facets were guideline, with 12 papers [38, 45, 20, 28, 16, 11, 47, 43, 44, 48–50], followed by lessons learned with 13 [26, 27, 29, 46, 30, 17, 31, 40, 32, 19, 34, 42, 36], then model facet with 5 [51, 39, 52, 12, 53], the framework facet with 5 [18, 15, 33, 35, 37], and finally advice with 2 articles [14, 41].

First of all, it is important to emphasize that the number of method found does not correspond to the number of articles (37 articles) since some articles use more than one method. The distribution of methods found in the review showed that most articles used the case study method (22 articles), followed by interviews (13 articles) and literature review (7 articles), and survey (5 articles). The least used methods were observations (3 articles) and exploratory research (1 article).

4.2. What Practices are used by GSD Projects that Employ such a Hybrid Approach?

Our SLR results in 37 included papers and we identified 16 practices that are adopted in a hybrid way by GSD projects (See Table 2). We use practice as a generic term to cover the way software is developed and its characteristics as well, which includes the software development, design, maintenance, and evolution phases. For instance, we can use the practice to refer to a well-established software development methodology (e.g., agile) or a specific way of adopting continuous integration during global software development.

Table 2. Practices by papers

Practices	Citations
Scrum of Scrums	[45, 18, 29, 14, 46, 16, 11, 40, 32, 15, 47, 43, 44, 48, 12]
Produce and track the quality plan.	[50, 20, 26, 51, 16, 39, 41, 19, 34, 42, 12]
Make the Process Flexible	[36, 20, 46, 17, 39, 31, 19, 12]
Necessary documentation	[53, 49, 37, 29, 16, 41, 52, 34]
Coordination Risks	[35, 38, 27, 16, 52, 33, 12]
Backlog Management	[37, 50, 38, 20, 39, 34]
Release planning	[50, 20, 30, 39, 44]
User Stories	[50, 20, 19, 44, 49]
Adapting agile team roles	[53, 35, 37, 46, 39]
End-to-End (System) Testing	[26, 29, 39, 44]
Formal estimation	[49, 28, 46, 17]
Expert/Team based estimation	[28, 46, 17]
Retrospectives	[38, 29, 11]
Sprint	[20, 29, 40]
Risk Mitigation	[35, 20, 39]
Limit Work-in-Progress	[38, 11]

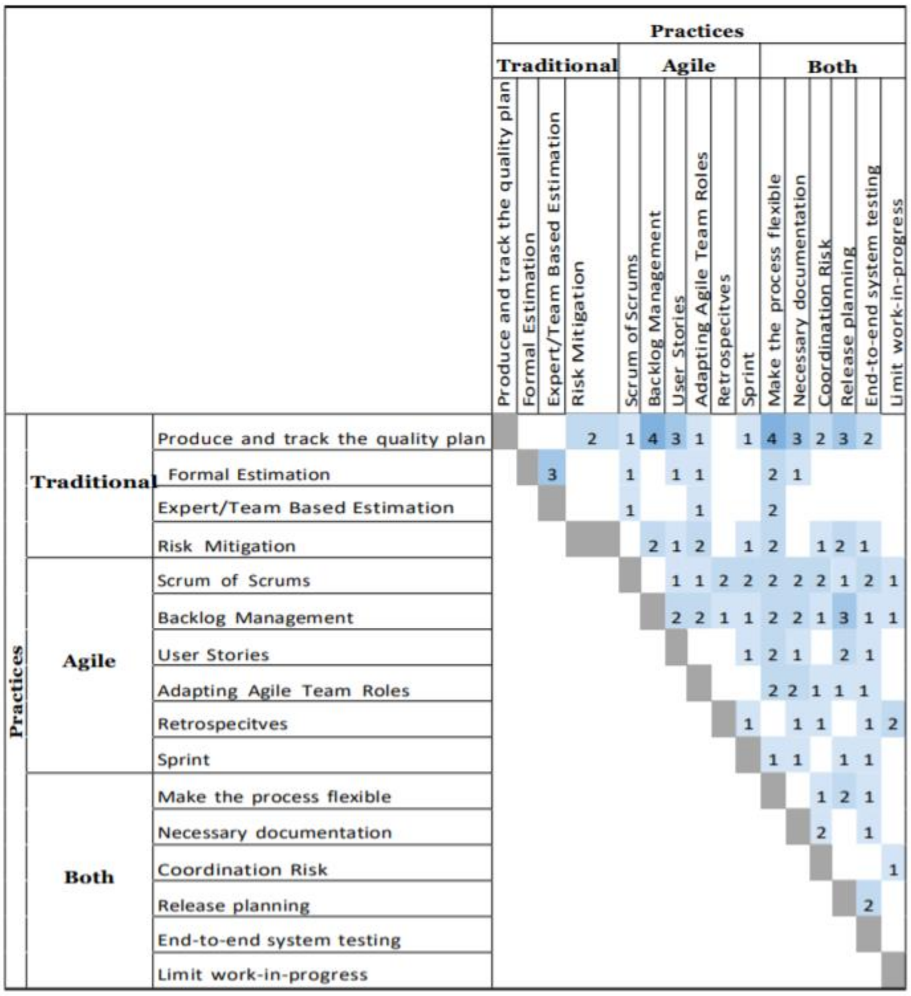
We also show in in Table 3 how each of the 16 practices are pairwise combined. The Scrum of Scrums practice 4.2 as the most used practice is also the most combined one.

To help implement our set of identified practices (See Table 2), we will describe each practice, in the next subsections, based on the review papers. The description aims at the goal of the practice, which should be applied to the practice, who should apply it, and how the practice can be adopted in a GSD environment. We emphasize that the practices presented in this work are contained in the HELENA Survey [54].

4.3. Scrum of Scrums

Many studies reported the use of Scrum of Scrums to better suit the agile ceremonies in a distributed environment [32, 15, 47, 43, 12]. The most common tailored agile framework was Scrum [45, 18, 29, 16, 11, 40, 44, 48, 14]. Goal: use the Scrum of Scrums to adapt agile ceremonies and events to make it suitable for a distributed environment. Who: dev teams, product owner, scrum master. How : some review studies stated that did not use standard user stories, but specified work items through conversation with developers [29]. Other reported adaptations on team roles, like a project and product manager on Scrum teams [14, 44]. Furthermore, an adaptation of the scrum meetings was presented, like having daily meetings twice-a- week, and asynchronous retrospective meetings [48]. Also, one study presented on its survey [46] agile distributed projects with teams up to 10 members, and some with more than 16 people or even 100 people.

Table 3. Overview of the pairwise combination of different hybrid practices.



			Practices															
			Traditional				Agile						Both					
			Produce and track the quality plan	Formal Estimation	Expert/Team Based Estimation	Risk Mitigation	Scrum of Scrums	Backlog Management	User Stories	Adapting Agile Team Roles	Retrospecitves	Sprint	Make the process flexible	Necessary documentation	Coordination Risk	Release planning	End-to-end system testing	Limit work-in-progress
Practices	Traditional	Produce and track the quality plan	▨			2	1	4	3	1		1	4	3	2	3	2	
		Formal Estimation		▨	3		1		1	1			2	1				
		Expert/Team Based Estimation			▨		1			1			2					
		Risk Mitigation				▨		2	1	2		1	2		1	2	1	
	Agile	Scrum of Scrums					▨		1	1	2	2	2	2	2	1	2	1
		Backlog Management						▨	2	2	1	1	2	2	1	3	1	1
		User Stories							▨			1	2	1		2	1	
		Adapting Agile Team Roles								▨			2	2	1	1	1	
		Retrospecitves									▨	1		1	1		1	2
		Sprint										▨	1	1		1	1	
	Both	Make the process flexible											▨		1	2	1	
		Necessary documentation												▨	2		1	
		Coordination Risk													▨			1
		Release planning														▨	2	
		End-to-end system testing															▨	
		Limit work-in-progress																▨

4.4. Produce and Track the Quality Plan

The quality plan development aims to ensure the quality of artefacts produced in the project. Since agile does not produce much documentation, it was seen as traditional documentation standards in reviewed studies [39]. Goal: ensure project quality by producing and tracking a quality plan during the process and activities conducted in the development. Who: team members. How : quality plans was seen through the documentation of critical artifacts to supplement informal communication, and also to formalize vendor-client relationship [26]. One study reported the use of the agile process combined with the delivery of required documents artifacts at major milestones to ensure project quality [41]. Furthermore, one article stated a quality approach that consists of documenting every evidence of testing and requirements implementation activities [19]. Also, it was reported by one article the elaboration of a quality plan adhering CMMI level 5 process [39]. Finally, one article encourages the proper documentation of components to ensure success during software integration [50].

4.5. Make the Process Flexible

The use of hybrid development process were present in some studies that combined agile with traditional methodologies, such as Rational Unified Process (RUP) [36, 20, 12], a sequential and linear model [39], and a plan- driven approach [17]. Goal : combine agile and traditional approaches to enhance the development process. Who: project manager, teams. How : Nuevo and Pino study [12] combined agile methods with RUP to perform project management that could suit a distributed environment, and they called it scRumUP. Furthermore, Sundararajan study et.al [39] presents a combination of the iterative and incremental model of agile to a linear model with sequential activities [39]. One study [31] reported the use of sequential development. In the process, each developer, in turn, will review the previous code written by a colleague, correct any errors found, and he/she will add their contribution of lines and transmit the code for the next developer to do the same [31].

4.6. Necessary Documentation

One agile principle states working software over documentation [7], although some distributed projects reported the need of documen- tation to plan and monitor the solution development [53, 29, 41]. Goal : document the necessary information regarding the project to help plan activities and monitor the project. Who: teams, product owner. How : one article suggests at least the documentation from a financial standpoint, it aims to cover the efforts and cost esti- mates to better perform a lifecycle planning [41]. Besides it, in a distributed project, one developer reported the documentation of big long documents of specifications [29], and one study suggests that business members should focus on documentation from the product team [16].

4.7. Coordination Risks

It aims to coordinate risks, to minimize the project’s misunderstanding, ambiguity and lack of communication [35, 38, 27, 33]. Goal : coordinate the project information through artefacts or documents to reduce risk regarding communication among teams. Who: teams. How : since in distributed projects, not all team members can be present in a meeting, to reduce risk regarding the information about these meetings one article pointed that the use of documentation can be valuable for teams distributed across different time zones [52]. Another article suggests the use of supplement documentation from meetings to help all members to follow the project activities [12]. Further, one article pointed to the coordination of risks through documents as a necessary practice to handle the process on every site of the GSD [16].

4.8. Backlog Management

The building of a project/product backlog was seen in reviewed studies as an opportunity to provide solution visibility across teams and to put on track its progress [37, 50, 38, 20, 34]. Goal :

development and management of a backlog to help coordination and visibility at the project. Who: project management, team leader. How: some studies suggest the development and management of the backlog based on the project plan [39, 34].

4.9. Release Planning

The release of software in agile teams usually occurs with constant frequency, although some reviewed studies stated the build of a software release plan [50, 20, 30, 39].Goal : development of a software release schedule that should be followed. Who: dev teams. How: one article reports the build of a software release plan and defined that releases would occur every three months [20], which is a hybrid approach that does not fully follow the agile practices of continuous integration and deployment that aim to release and integrate software frequently. However, it is more similar to a traditional approach that defines specific milestones for software releases.

4.10. User Stories (as Requirements Engineering Practice)

Requirements specification is used to describe the desired behavior of a system through a formal and detailed specification. In agile, requirements are described as user stories, although, formal specifications were seen in GSD projects [50, 20, 19, 44, 49]. Goal: describe system behavior through a documented specification. Who: team members. How: studies do not fully describe how to perform requirements specification. Otherwise, in the context of GSD, one article reported that specifications should be written for everything that is implemented [19]. We found that in GSD teams, some stud- ies pointed to the necessity of requirements documentation to make maintenance easier since the developers could consult the expected system behavior [49, 37, 52]. Furthermore, one article reported the documentation of use cases [34].

4.11. Adapting Agile Team Roles

In agile the common team roles are product owner, scrum master, and the team [55]. However, some reviewed studies present the adaptation of those roles and the creation of new ones [53, 35, 37, 46, 39]. Goal : adapt the agile team roles and create new ones to suit the distributed environment. Who: project manager. How : a study reported the existence of different roles in the project, such as architect, designer, business analyst, test specialist, onsite coordinator, build tracker, daily progress tracker, and moderator [39].

4.12. End-to-End (System) Testing

In agile, test activities are usually conducted during development by the teams, in review by the product owner, and system demos by the customer. However, in some studies, the presence of end-to-end tests to ensure quality was not even executed [29], and in others, a dedicated QA team was present to ensure the quality of the increments developed [26]. Goal: it aims to maintain quality control of the process and the product. Who: QA team, dev team. How: one study reported the presence of a dedicated QA team, which at each iteration of the project was responsible to review the test procedures executed by an offshore team to ensure quality [26]. Besides it, the QA team was responsible to evaluate whether the practices of the offshore team were of acceptable quality. Finally, a study reported a traditional technique of executing system and integration tests only before a release to measure technical performance, functional validation, and quality of the product [39].

4.13. Formal estimation

It measures the software features to be developed, team performance, and development flow [46]. It aims to quantify the features that are being developed, the team effectiveness to understand how much value they are delivering to the customer, and also measure the performance of the team to keep balance in the development flow [28, 46, 17]. Goal: measure the development process to improve it, keep the balance of team performance, and also define more precise estimations. Who: dev teams. How : in agile the story points estimation metric is commonly used, although, some studies have reported the use of traditional estimation techniques global distributed projects, such as functions points and lines of code [46, 17], use case points [28, 46], COSMIC [46], estimation by analogy [28], case base reasoning [17], Delphi, UML points and COCOMO [46, 17]. The use case points count was also used [28, 46] and some cases were customized, estimating test effort only [17]. One article also reported as a mitigation strategy to the challenges of requirements re-usability the use of reuse metrics, in order, to quantify the number of re-used requirements in the project [49]. Finally, one study presented the use of traditional accuracy metrics in GSD projects, such as the magnitude of the relative error (MRE) or variation to assess the accuracy of their estimation techniques [17].

4.14. Expert/Team Based Estimation

It can give a quick and approximate estimate of how much work is expected for a project. The sizes can be converted into numbers at a later stage – when the team assigns a relative size to the project on hand [46, 17]. Goal : This is decided by dialogue and collaborative works to agree with everything that allow the teams to estimate the necessary effort to develop new features in the project [46, 17].Who: dev teams. How : in agile, planing poker is the most popular estimation method, although, in our findings, a traditional estimation technique was used in globally distributed teams as expert judgment [28, 46, 17].

4.15. Retrospectives

Retrospective is a mainly agile event, although some studies reported the use of it at different intervals than the agile practice [29, 11]. Goal : check and improve the solution development process. Who: dev teams and scrum master. How : in one of the studies a team member stated that retrospective meetings were rarely held, he argued that only one meeting was done in two years, and it was after a release, not a sprint [29]. Furthermore, another study reported holding a retrospective meeting monthly after two sprints [11].

4.16. Sprint

In Scrum, sprints are time boxed events in which the teams commit them- selves to the development of new features and no changes are allowed during the period [55], although in some reviewed studies, changes in the sprint backlog hap- pened and the sprint duration changed [20, 29, 40]. Goal : implementation of changes during the sprint. Who: scrum master and product owner. How : one study reported that a product owner often altered the requirements in the middle of the sprint, and as a consequence, the scrum master changed the sprint duration or delayed the next release [40]. One study also suggests the development of upfront design models to provide visibility for all teams since the beginning, like a project plan [20]. However, those models are not flexible for changes, and because of it, they can become frequently outdated if the project vision change due to new requirements or new prioritization [20].

4.17. Risk Mitigation

It is a strategy to prepare, and evaluate the effects of threats faced by a project [39]. In agile, continuous feedback, review with the customer, and the management of changes help to reduce risks [7]. However, it was seen in GSD projects some traditional methods to mitigate risks [35, 39].Goal : define a set of actions to mitigate the damage caused by possible risks regarding the project. Who: team members. How : an article reported the use of a traditional framework to manage risk, the GDSP from Persson [39]. The same study reported the use of other traditional techniques to mitigate risk, such as follow the CMMI level 5 process and use UML documentation guidelines [39]. Other study suggest the the conduction of vision and roadmap planning to align all team members with the project vision and milestones, although, such study did not consider the presence of the end-users and stakeholders [20].

4.18. Limit Work-in-Progress (WIP)

Limit the work in progress is characterized by the definition of limits for activities in development Goal : aggregate and execute the maximum of work items, and also establish a limit for it.Who: dev teams, product owner, scrum master. How : one study reported the execution of planning once a month, but it increased the amount of work and take too much time to make all the teams understand the tasks and synchronize their work hours [38]. Furthermore, one distributed project reported the use of a general Kanban board with WIP limits to cover all three teams' activities [11].

5. DISCUSSION

Even with several studies presenting successful implementations of agile method- ologies in different contexts, [2, 19], most of those methods are not implemented in their "purest form" [9]. Further, it has been a challenge to perform them in environments with distributed teams developing large software solutions [3, 14]. Also, Contractual items, for projects under development for external organizations, introduce additional complexities for purely agile-based approaches.

Coordinating software development with teams dispersed over various sites can be challenging. The use of agile practices in distributed projects seems to be a good approach to coordinate the development process [56]. However, in such complex environments with teams distributed across the globe, the use of traditional practices also seems to be suitable [57]. The combination of both agile and traditional practices generates hybrid practices that aim to help the development and coordination of distributed projects, and such practices are being constantly used by companies with globally distributed settings.

Software development practices perform an essential role in the final software product quality [58]. From the 16 hybrid practices extracted during the SLR, the most cited was Scrum of Scrums 4.2, which focused on the adaption of the Scrum framework to handle the complexity of global distribution teams. The practice has shown the use of "daily meetings" twice-a-week [48]. Such a manner to conduct daily meetings is not present in any agile methodology, which defines this approach as an agile adapted one, although the adaptation could be helpful since distributed teams have difficulty synchronizing their work hours across different timezones. Besides it, some distributed teams reported the use of software release plans 4.2 with releases every three months [20]. Such practice aims to organize and define the release of new versions to better serve the customer, although it uses a plan-driven approach of releasing software in a large scheduled approach, different from agile that reinforces the use of continuous deployment and integration.

Some hybrid practices were related to the requirements engineering area, such as the User Stories practice 4.2, the Formal Estimation 4.2 and Expert/Team based estimation 4.2 practices. In those practices, the use of traditional practices was predominant, such as use case points, function points [46, 17], expert judgment, COCOMO, UML points, estimation by analogy, etc [28, 46, 17]. Such practices confirm the use of traditional techniques depending on the complexity of the project. Furthermore, requirements documentation was reported by some studies that stated the need to document everything that was developed [19] or just to document the requirements to make all teams aware that what has been developed. Such practice is not present in agile since the use of user stories is encouraged without the need to document. However, document and make information available to anyone in the project can reduce misunderstandings regarding features developed, especially in GSD projects [49, 37].

We can also see hybrid practices that present the use of CMMI standards due to customer contracts [39], project plans to mitigate risks [39], and hybrid development approaches that combined agile methods with RUP [20, 12], plan-driven [17], and waterfall model [18, 19] to better manage and coordinate the distributed project. Furthermore, one study identified on the sprint practice 4.2 reported the use of upfront design models [20], which could help the team with visibility about the project, although it disables the process to be flexible to changes, which could cause issues in a distributed context.

It is also possible to point that some hybrid practices were used by GSD teams due to some context factors pointed by Klunder et al. [8] in their study. Context factors such as the distribution of development and application domain could influence the development method and consequently the development practices used by the GSD teams. Practices like backlog management 4.2, make the process flexible 4.2, necessary documentation 4.2, Scrum of Scrums 4.2, and others are an example of practices that were applied by GSD teams due to distribution, and the need to scale the process.

Similar to what was presented in the Scott et al. [13] study, the Scrum framework was also the most used in the GSD studies [59]. Such a fact can also be related to a finding of the HELENA study [54], which states that even with standards, norms, and regulated domains, the companies are used to adopt agile methods. However, the agile practices were mostly of the time adapted to be suitable for the distributed contexts [32, 15, 47, 43]. Such changes in the agile practices also reinforce that is possible to adapt them to specific scenarios and also be benefited from them to achieve success [29, 14, 44, 48].

It is possible to notice that the adoption of hybrid practices such as Scrum of Scrums 4.2, End-to-End (System) Testing 4.2 and Formal Estimation 4.2 enable the improvement of the development process. Such practices make it possible to achieve the main objectives pointed out by Marinho et al et al. [4] and Jil Klunder [10], such as, increased productivity, better product quality, better project planning, and increased frequency of deliveries to the customer.

Make the process flexible to achieve a combination of traditional and agile methodologies, seems to be a good approach in some studies [20, 12]. Nuevo et al.

[12] presented an approach originated from the combination of RUP and Scrum, which was helpful to deal with the challenges of GSD, and also was flexible for adaptations according to the needs of the project. On another side, Cho et al. [60] published a hybrid software development method also based on Scrum and RUP. However, Cho's method was more structured, making the scrum practices in the center of the method and keeping the RUP structure in the architecture of the model, resulting in a more solid method, but less flexible. We believe that both approaches

could bring benefits for large-scale and global projects, although, in high challenging scenarios, it is preferable to use hybrid practices that can deal better with changes.

Looking at the HELENA study [54], it is possible to make a relation between the practices reported in the HELENA and the practices presented on this SLR. More than half of the practices extracted from the SLR are also present in the HELENA study [54]. From the 35 hybrid practices identified in the HELENA study, 9 of them are in use in global software development teams, according to our SLR. Those 9 practices are Scrum of Scrums (4.2), Backlog Management (4.2), Release Planning (4.2), User stories (4.2), End-to-End (System) Testing (4.2), Formal estimation (4.2), Expert/Team based estimation (4.2), Retrospectives (4.2), Limit Work-in-Progress (WIP) (4.2). Such relation reinforce the SLR results, which has shown real hybrid practices used by GSD projects that are also presented in the HELENA study.

According to Tell et al. [21] the standard software engineering activities, such as testing, risk mitigation, and reviews are executed by most of the software development approaches. Further, such standard activities have not changed due to the emerging of the agile methods, actually which changed is the way that those activities are used and combined. Based on the hybrid practices extracted on this SLR, we can support this finding.

Some SLR studies that reported GSD projects have shown that some of them are not fully agile, due to the use of hybrid practices to manage distributed and complex environments [45, 46]. The agile approach suggests light and thin methods, although it cannot avoid dealing with high-level complexity in distributed projects. Due to it, the use of hybrid practices are present in distributed projects since it allows alignment among teams, adapt the development process to serve the customer and teams, make information available to anyone in the project through documentation, monitor and measure the project flow through quality plans and risk mitigation techniques for consequently achieve the project success.

5.1. Threats To Validity

Construct Validity to ensure this validity, the design, data extraction, and analysis of the SLR study were conducted through a strict and stable research protocol developed, discussed, and validated by the authors to ensure reliability. Also, all authors discussed each paper selected, and practice extracted, and if any of them disagreed with the choices made, they would discuss to reach a consensus. Such validity can also be related to the creation of the search string, although to mitigate such threat we added synonyms for different words of the search string, e.g hybrid development.

Internal Validity the search strategy was developed by the three research students and reviewed by the supervisor. Although we only searched five online digital libraries, they are believed to cover the majority of the high quality publications in Software Engineering.

External Validity we selected studies that include a discussion about hybrids practices in GSD from 2001 to 2020. The excluded papers without hybrids practices in GSD issues reported may affect the generalizability of our result. Furthermore, hybrid practices usually emerge in real scenarios due to circumstances faced by the companies to achieve a software development process that matches the teams, the project, the customers [9]. However, an SLR approach was chosen for this study, such a method allows the authors to evaluate the state of the art and primary studies regarding their need to apply hybrid development practices. Also, the practices presented through the SLR are aligned in name and definition with the HELENA survey [54] and other papers in the same research area [21, 4]. Due to it, the practices seen in the study make with what is being researched.

Conclusion validity we extracted the data from the selected papers with Hybrid practices in the GSD context. To ensure the truth of the extracted data, a research protocol was developed to describe the data extraction strategy and format. We defined a data extraction form to obtain uniform extraction of relevant information and indicated whether the data to be extracted would address the research questions. Besides, the crosscheck was necessary among the reviewers, and we had at least two researchers extracting data independently. The supervisor dealt with any divergences and disagreements during the process.

6. CONCLUSION

In this study, we investigated and analyzed the GSD literature to identify teams that were using hybrid practices to maintain and conduct their projects. Through an SLR with 37 articles selected, we could identify 16 hybrid practices applied in GSD teams. Furthermore, we presented these 16 practices into a structure of description, goal of the practice, who should apply it, and how it could be applied.

Our systematic literature review on hybrid software development in the global software development context provided a set of evidence related to hybrid practices. It was used five renowned bibliographic databases (IEEEXplore, ACM Digital Library, SpringerLink, Scopus, and Wiley), which allowed us to find articles with relevant content regarding GSD and hybrid software development practices. Some of the most used hybrid practices were Scrum of Scrums, Produce and track the quality plan, Make the process flexible, and Necessary documentation.

The use of hybrid practices emerges from the combination of agile practices and traditional practices, in order, to achieve the best of both worlds in the software development context. Furthermore, the use of hybrid approaches in global software development settings is not different, most of the projects use it to reach a good performance level, and to deal with many challenges regarding cultural barriers, time zone differences, and geographic boundaries. In this study, we contributed to both practitioners and researchers providing a set of hybrid practices used in GSD contexts and describing how to use it, who should use it, and the goal of each.

Finally, as future work, we intend to investigate the effectiveness, the impact, and the mix-match combinations of the hybrid practices in GSD settings. Also, we aim to extract the usefulness of adopting hybrid approaches and which combinations are the most suited for distributed contexts.

REFERENCES

[1] M. Marinho, J. Noll, and S. Beecham, “Uncertainty management for global software devel- opment teams,” in 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC). Coimbra, Portugal: IEEE, Sep. 2018, pp. 238–246.

[2] M. Paasivaara and C. Lassenius, Using Scrum Practices in GSD Projects. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 259–278.

[3] P. Lous, P. Tell, C. B. Michelsen, Y. Dittrich, and A. Ebdrup, “From scrum to agile: A journey to tackle the challenges of distributed development in an agile team,” in Proceedings of the 2018 International Conference on Software and System Process, ser. ICSSP ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 11–20.

[4] 4. M. L. Marinho, J. Noll, I. Richardson, and S. Beecham, “Plan-driven approaches are alive and kicking in agile global software development,” in International Symposium on Empirical Software Engineering and Measurement (ESEM). Porto de Galinhas, Brazil: IEEE, 09 2019, pp. 1–11.

[5] M. Marinho, A. Luna, and S. Beecham, “Global software development: practices for cultural differences,” in International Conference on Product-Focused Software Process Improvement. Springer, 2018, pp. 299–317.

[6] P. Tell, J. Klu¨nder, S. Ku¨pper, D. Raffo, S. MacDonell, J. Mu¨nch, D. Pfahl, O. Linssen, and M. Kuhrmann, “Towards the statistical construction of hybrid development methods,” Journal of Software: Evolution and Process, vol. 33, p. e2315, 2020.

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, “Manifesto for agile software development,” 2001. [Online]. Available: http://www.agilemanifesto.org/

[8] J. Klu¨nder, D. Karajic, P. Tell, O. Karras, C. Mu¨nkel, J. Mu¨nch, S. G. MacDonell, R. Hebig, and M. Kuhrmann, Determining Context Factors for Hybrid Development Methods with Trained Models. New York, NY, USA: Association for Computing Machinery, 2020, p. 61–70.

[9] M. Kuhrmann, P. Diebold, J. Mu¨nch, P. Tell, V. Garousi, M. Felderer, K. Trektere, F. McCaffery, O. Linssen, E. Hanser, and C. R. Prause, “Hybrid software and system development in practice: Waterfall, scrum, and beyond,” in Proceedings of the 2017 International Conference on Software and System Process, ser. ICSSP 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 30–39.

[10] J. Klu¨nder, P. Hohl, M. Fazal-Baqaie, S. Krusche, S. Ku¨pper, O. Linssen, and C. R. Prause, “Helena study: Reasons for combining agile and traditional software development approaches in german companies,” in Product-Focused Software Process Improvement. Cham: Springer International Publishing, 2017, pp. 428–434.

[11] R. Vallon, K. Bayrhammer, S. Strobl, M. Bernhart, and T. Grechenig, “Identifying critical areas for improvement in agile multi-site co-development,” ENASE 2013 - Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering, vol. 1, pp. 165–172, 01 2013.

[12] E. d. Nuevo, M. Piattini, and F. J. Pino, “Scrum-based methodology for distributed software development,” in 2011 IEEE Sixth International Conference on Global Software Engineering. Helsinki, Finland: IEEE, 2011, pp. 66–74.

[13] E. Scott, D. Pfahl, R. Hebig, R. Heldal, and E. Knauss, “Initial results of the helena survey conducted in estonia with comparison to results from sweden and worldwide,” in Product- Focused Software Process Improvement. Cham: Springer International Publishing, 2017, pp. 404–412.

[14] R. K. Gupta and P. Manikreddy, “Challenges in adapting scrum in legacy global configurator project,” in 2015 IEEE 10th International Conference on Global Software Engineering. Ciudad Real, Spain: IEEE, 2015, pp. 46–50.

[15] B. Ramesh, K. Mohan, and L. Cao, “Ambidexterity in agile distributed development: An empirical investigation,” Info. Sys. Research, vol. 23, no. 2, p. 323–339, Jun. 2012.

[16] B. Rizvi, E. Bagheri, and D. Gasevic, “A systematic review of distributed agile software engineering,” Journal of Software: Evolution and Process, vol. 27, 06 2015.

[17] R. Britto, M. Usman, and E. Mendes, “Effort estimation in agile global software development context,” in Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation. Cham: Springer International Publishing, 2014, pp. 182–192.

[18] O. Sievi-Korte, I. Richardson, and S. Beecham, “Software architecture design in global software development: An empirical study,” Journal of Systems and Software, vol. 158, p. 110400, 08 2019.

[19] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in a globally distributed project: A case study,” Softw. Process, vol. 13, no. 6, p. 527–544, Nov. 2008.

[20] R. Lal and T. Clear, “Enhancing product and service capability through scaling agility in a global software vendor environment,” in Proceedings of the 13th International Conference on Global Software Engineering, ser. ICGSE ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 59–68.

[21] P. Tell, J. Klu¨nder, S. Ku¨pper, D. Raffo, S. G. MacDonell, J. Mu¨nch, D. Pfahl, O. Linssen, and M. Kuhrmann, “What are hybrid development methods made of? an evidence-based characterization,” in Proceedings of the International Conference on Software and System Processes, ser. ICSSP ’19. Montreal, Quebec, Canada: IEEE Press, 2019, p. 105–114. [Online]. Available: https://doi.org/10.1109/ICSSP.2019.00022

[22] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” Technical report, EBSE Technical Report EBSE-2007-01, Tech. Rep., 2007.

[23] C. Wohlin, P. Runeson, M. Ho¨st, M. Ohlsson, B. Regnell, and A. Wessl´en, Experimentation in Software Engineering. Midtown Manhattan, New York City: Springer, 2012, pp. 123–151.

[24] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, ser. EASE'08. Swindon, GBR: BCS Learning & Development Ltd., 2008, p. 68–77.

[25] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classifica- tion and evaluation criteria: A proposal and a discussion," Requir. Eng., vol. 11, pp. 102–107, 03 2006.

[26] B. Ramesh, L. Cao, K. Mohan, and P. Xu, "Can distributed software development be agile?" Commun. ACM, vol. 49, no. 10, p. 41–46, Oct. 2006.

[27] I. Richter, F. Raith, and M. Weber, "Problems in agile global software engineering projects especially within traditionally organised corporations: [an exploratory semi-structured interview study]," in Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering, ser. C3S2E '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 33–43.

[28] M. Usman and R. Britto, "Effort estimation in co-located and globally distributed agile software development: A comparative study," in 2016 joint conference of the international workshop on software measurement and the international conference on software process and product measurement, IEEE. Berlin, Germany: IEEE, 2016, pp. 219–224.

[29] M. A. Razzak, J. Noll, I. Richardson, C. N. Canna, and S. Beecham, "Transition from plan driven to safe®: Periodic team self-assessment," in Product-Focused Software Process Improvement. Cham: Springer International Publishing, 2017, pp. 573–585.

[30] N. Tripathi, P. Rodr´ıguez, M. O. Ahmad, and M. Oivo, "Scaling kanban for software development in a multisite organization: Challenges and potential solutions," in Agile Processes in Software Engineering and Extreme Programming. Cham: Springer International Publishing, 2015, pp. 178–190.

[31] A. M. E. Hamid, "Upgrading distributed agile development," in 2013 Internatonal Conference on Computing, electrical and eletronic engineering (ICCEEE). Khartoum, Sudan: IEEE, 2013, pp. 709–714.

[32] M. Tanner and W. Chigona, "Towards an understanding of the contextual influences on dis- tributed agile software development: A theory of practice perspective," ECIS 2012 - Proceedings of the 20th European Conference on Information Systems, vol. 178, 01 2012.

[33] E. Hossain, M. Ali Babar, and J. Verner, "Towards a framework for using agile approaches in global software development," in Product-Focused Software Process Improvement. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 126–140.

[34] S. Hole and N. B. Moe, "A case study of coordination in distributed agile software development," in Software Process Improvement. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 189–200.

[35] M. Esteki, T. J. Gandomani, and H. K. Farsani, "A risk management framework for distributed scrum using prince2 methodology," Bulletin of Electrical Engineering and Informatics, vol. 9, no. 3, pp. 1299–1310, 2020.

[36] R. Akbar, A. R. Khan, and K. Adnan, "Software development process evolution in malaysian companies," in Data Management, Analytics and Innovation. Singapore: Springer, 2020, pp. 139–150.

[37] S. McCarthy, P. O'Raghallaigh, C. Fitzgerald, and F. Adam, "Towards a framework for shared understanding and shared commitment in agile distributed isd project teams," in ECIS 2019, Proceedings of the 27th European Conference on Information Systems, AIS Electronic Library (AISeL). Stockholm & Uppsala, Sweden: Association for Information Systems AIS Electronic Library (AISeL), 2019, pp. 1–15.

[38] D. M. Szabo´ and J. Stegho¨fer, "Coping strategies for temporal, geographical and sociocultural distances in agile gsd: A case study," in 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). Montreal, QC, Canada, Canada: IEE, 2019, pp. 161–170.

[39] S. Sundararajan, M. Bhasi, and P. K. Vijayaraghavan, "Case study on risk management practice in large offshore-outsourced agile software projects," IET Software, vol. 8, no. 6, pp. 245–257, 2014.

[40] P. Ralph and P. Shportun, "Scrum abandonment in distributed teams: A revelatory case," in Proceedings - Pacific Asia Conference on Information Systems, PACIS 2013, vol. 101. Jeju Island, Korea: AIS, 06 2013, pp. 1–16.

[41] A. Avritzer, F. Bronsard, and G. Matos, Improving Global Development Using Agile. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 133–148.

[42] C. Kussmaul, R. Jack, and B. Sponsler, "Outsourcing and offshoring with agility: A case study," in Extreme Programming and Agile Methods - XP/Agile Universe 2004. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 147–154.

[43] T. Dingsøyr, N. Moe, T. Fægri, and E. Amdahl Seim, “Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation,” Empirical Software Engineering, vol. 23, pp. 1–31, 06 2017.

[44] M. M. Jha, R. M. F. Vilardell, and J. Narayan, “Scaling agile scrum software development: Providing agility and quality to platform development by reducing time to market,” in 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE). Irvine, CA, USA: IEEE, 2016, pp. 84–88.

[45] P. Lous, M. Kuhrmann, and P. Tell, “Is scrum fit for global software engineering?” in Proceedings of the 12th International Conference on Global Software Engineering, ser. ICGSE ’17. Buenos Aires, Argentina: IEEE Press, 2017, p. 1–10.

[46] R. Britto, E. Mendes, and J. Bo¨rstler, “An empirical investigation on effort estimation in agile global software development,” in 2015 IEEE 10th International Conference on Global Software Engineering. Ciudad Real, Spain: IEEE, 2015, pp. 38–45.

[47] J. Bass, “How product owner teams scale agile methods to large distributed enterprises,” Empirical Software Engineering, vol. 20, pp. 1525–1557, 12 2015.

[48] E. Hossain, M. A. Babar, and H. Paik, “Using scrum in global software development: A systematic literature review,” in 2009 Fourth IEEE International Conference on Global Software Engineering. Limerick, Ireland: IEEE, 2009, pp. 175–184.

[49] S. S. Hossain, Y. Arafat, T. Amin, and T. Bhuiyan, “Requirements re-usability in global software development: A systematic mapping study,” in International Conference on Computational Science and Its Applications, Springer. Cagliari, Italy: Springer, Cham, 2020, pp. 960–974.

[50] M. Ilyas, S. U. Khan, and N. Rashid, “Empirical validation of software integration practices in global software development,” SN Computer Science, vol. 1, pp. 1–23, 2020.

[51] M. Kausar and A. Al-Yasiri, Using Distributed Agile Patterns for Supporting the Requirements Engineering Process. Cham: Springer International Publishing, 2017, pp. 291–316.

[52] A. Ansari, S. Sharafi, and N. Nematbakhsh, “A method for requirements management in distributed extreme programming environment,” Journal of Theoretical and Applied Information Technology, vol. 20, pp. 52–58, 10 2010.

[53] Y. I. Alzoubi and A. Q. Gill, “An empirical investigation of geographically distributed agile development: The agile enterprise architecture is a communication enabler,” IEEE Access, vol. 8, pp. 80 269–80 289, 2020.

[54] M. Kuhrmann, P. Tell, J. Klu¨nder, R. Hebig, S. Licorish, and S. MacDonell, “Helena stage 2 results,” 11 2018.

[55] K. Schwaber and M. Beedle, Agile Software Development with Scrum, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[56] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in distributed agile development: A multiple case study,” in 2009 Fourth IEEE International Conference on Global Software Engineering. Limerick, Ireland: IEEE, 2009, pp. 195–204.

[57] G. Theocharis, M. Kuhrmann, J. Mu¨nch, and P. Diebold, “Is water-scrum-fall reality? on the use of agile and traditional development practices,” in Product-Focused Software Process Improvement. Cham: Springer International Publishing, 2015, pp. 149–166.

[58] M. Viggiato, J. Oliveira, E. Figueiredo, P. Jamshidi, and C. Ka¨stner, “Understanding similarities and differences in software development practices across domains,” in 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), IEEE. Montreal, QC, Canada: IEEE, 2019, pp. 84–94.

[59] R. Camara, A. Alves, I. Monte, and M. Marinho, “Agile global software development: A systematic literature review,” in Proceedings of the 34th Brazilian Symposium on Software Engineering, ser. SBES ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–40. [Online]. Available: https://doi.org/10.1145/3422392.3422411

[60] J. Cho, “A hybrid software development method for large-scale projects: rational unified process with scrum,” Journal of Issues in Information Systems, vol. 5, no. 2, pp. 340–348, 2009.