Universidad Popular Autónoma del Estado de Puebla
Centro Interdisciplinario de Posgrados
Investigación y Consultoría
Departamento de Ingeniería
Doctorado en Planeación Estratégica y Dirección de Tecnología

Título de la investigación

Strategic analysis of effort estimation considering the context in the agile development of software projects

Tesis que para obtener el Grado de Doctora

en Planeación Estratégica y Dirección de Tecnología

Presenta

María Guadalupe Medina Barrera

UPAEP – Secretaría General
Dirección General de Apoyos Académicos
Dirección del Centro de Recursos para el Aprendizaje y la Investigación.
Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Universidad Popular Autónoma del Estado de Puebla

Decanato de Ingeniería

Doctorate in Strategic Planning
and Technology Management

We hereby certify that this doctoral thesis has been approved:

Strategic analysis of effort estimation considering the context in the agile development of software projects

María Guadalupe Medina Barrera

Doctoral Committee

Dra. Rosa María Cantón Croda
Thesis Director

Dr. Damián Emilio Gibaja Romero
Thesis Co-Director

Dra. Alejandra Aldrette Malacara
Advisor 1

Dr. Juan Carlos Pérez García
Advisor 2

Dra. Argelia Fabiola Miranda Pérez
Advisor 3

Puebla, México. February, 2022.

Acknowledgements

I want to thank the Tecnológico Nacional de México for granting me a commission-license with which it was possible for me to carry out my doctoral studies. I also thank the directors and management personnel of the Instituto Tecnológico de Apizaco for supporting me by accomplishing the required procedures.

I also want to thank Universidad Popular Autónoma del Estado de Puebla as my doctoral study house, as well as the members of the doctoral faculty of the doctoral program in Strategic Planning and Technology Management, who shared their wisdom and experience with me during the various classes, seminars and colloquiums.

I want to express my gratitude to my directors of thesis, Dra. Rosa María Cantón Croda and Dr. Damián Emilio Gibaja Romero, who guided and support me throughout this research. Thanks a lot for giving me your invaluable support, knowledge, experience and advice.

My thanks to my thesis advisor, Dra. Alejandra Aldrette Malacara, whose advice and feedback allowed me to improve this thesis.

My special thanks to my husband José Juan for being with me at all times and always making me feel his support with love. Thanks love.

I want to thank with great love my family who has always supported me in everything I have undertaken in life, my deep gratitude for always encouraging me to move forward. Infinite thanks to you, Mom, Grandma, Lore, Willy, Candita. Above all, thank you God for blessing me.

Contents

Abstract viii

List of Figures ix

List of Tables xi

Chapter 1 Introduction 1

1.1 Problem statement 1

1.2 Research objectives 3

1.3 Methodology 4

1.4 Organization of the thesis 4

Chapter 2 Complexity of strategic analysis to estimate effort 7

2.1 The strategy 7

2.2 Strategic analysis of effort estimation 8

2.2.1 Effort and game theory 9

2.2.2 The agency theory in software development 12

2.2.3 Moral hazard in effort estimation 14

2.2.4 The teamwork quality impact 15

2.3 Effort estimation in agile software development 16

2.3.1 Software projects 16

2.3.2 Global software development 16

2.3.3 Agile software development 18

2.3.4 The effort estimation process 20

2.3.5 The Top-Down strategy . . . 21
2.3.6 The Bottom-Up strategy . . . 23

Chapter 3 EE in agile software development with insourcing mature teams 25

3.1 Introduction . . . 25
3.2 Product planning in Scrum . . . 28
3.3 The model . . . 29
3.3.1 Basics . . . 29
3.3.2 Game description . . . 30
3.3.3 The solution concept . . . 31
3.4 Equilibrium . . . 32
3.4.1 Payoff functions . . . 32
3.4.2 DT behaviour at equilibrium . . . 34
3.4.3 Resources at stage 1 equilibrium . . . 35
3.5 Interaction's variations . . . 37
3.5.1 Scenario 1 . . . 37
3.5.2 Scenario 2 . . . 40
3.5.3 Comparison between scenarios' equilibrium . . . 44
3.6 Numerical examples . . . 49
3.6.1 Changes in DT's strategy and payoff . . . 49
3.6.2 Changes in SM's strategy and payoff . . . 51
3.7 Conclusions . . . 52

Chapter 4 EE in agile global software development with offshoring teams 57

4.1 Introduction . . . 57
4.2 Effort estimation in agile global software development . . . 59
4.3 The model . . . 60
4.3.1 The game . . . 61
4.3.2 The solution concept . . . 63
4.4 The Bayesian Equilibria . . . 63

4.4.1 The DT's optimal decision rule 63
4.4.2 The support at equilibrium 65
4.4.3 Analysis of equilibria 69
4.5 Numerical examples 71
4.6 Conclusions 75

Chapter 5 Conclusions **79**
5.1 Conclusions 79
5.2 Future work 81

Appendix **83**
Appendix A 83
Appendix B 86

Bibliography **88**

Abstract

The present thesis addresses effort estimation (EE) of software projects in agile environments as a strategic model following a game-theoretical approach. This is the main contibution of this research. Although there is a significant number of techniques and methodologies to EE, the top-down and bottom-up perspectives depend mutually when making estimates. The strategic analysis of EE allows finding the optimal solution to achieve the project objectives in time and costs. In Scrum, the scrum master and the development team make decisions strategically as their goals depend on each other. Thus, determining EE considering the strategic behavior of the parties involved contributes to innovation in software projects management.

Based on Scrum ceremonies, we analyze the impact of interactions between the scrum master and the team during the EE process. To compute players' payoff, we consider the contextual factors that literature states as specific to agile software development (SD) environments. Then, we find the optimal strategies on effort and resources. This embrace a systemic view of EE strategic analysis to understand player strategies since they need coping with the impact of change during SD.

We rely on agency theory to investigate both the complete and incomplete information cases. Regarding the former, we study the case where the parties know each other's capabilities, as occurs with the insourcing SD with mature teams. Next, we address the latter, we analyse the case where the scrum master has limited information about team's skills which concerns to offshoring squemes in global SD. Findings show that EE stays the same around Scrum meetings, while player benefits and leader strategy change. Furthermore, we observe that exogenous factors (environmental stability and *sprint backlog* complexity) always impact on players' strategies and profits. Interestingly, as stability in global

SD increases, medium level of complexity increases both strategies and payoffs, while low complexity decreases them. Above all, findings reveal the importance of the scrum master's leadership style on software project outcomes; specially when retrospective meetings take place.

Keywords: Effort estimation, Agile Software Development, Game Theory, Insourcing Mature Teams, Offshore Teams, Servant Leadership.

List of Figures

2.1 The prisoner's dilemma framework. 10

2.2 Sourcing strategies of team members in GSD. 17

2.3 The framework Scrum. 19

3.1 The sequence of events around the EE game under the Scrum context. 28

3.2 Effort estimation game. 31

3.3 Scenario 1. 38

3.4 Scenario 2. 41

3.5 Changes in ϵ^* and π_{DT} as h, f, γ and τ change. 50

3.6 Changes in r^* and π_{SM} as h, f, γ and τ change. 52

4.1 The sequence of events around a Distributed EE setting. 60

4.2 Offshoring EE game. 61

4.3 Example of *SM* expected utility function. 68

4.4 Changes in equilibrium strategies and payoffs as τ and γ change with different initial amounts of communication investment. 73

4.5 Changes in equilibrium strategies and payoffs as τ and γ change with different initial amounts of SD resources. 75

List of Tables

3.1 Model Parameters 34

3.2 Values assigned to exogenous variables 50

3.3 DT's strategy and payoffs as h, f, γ and τ change 51

3.4 SM's strategy and payoffs as h, f, γ and τ change 53

4.1 Values assigned to exogenous variables 71

4.2 Equilibrium strategies and payoffs as τ and γ change with different initial amounts of communication investment 72

4.3 Equilibrium strategies and payoffs as τ and γ change with different initial amounts of SD resources 74

Chapter 1

Introduction

1.1 Problem statement

The software is expected to combine with other industries to address the greatest needs of human beings, impacting almost all social and economic sectors (Betz et al., 2019). Moreover, the global trend towards digitalization has increased with the current pandemic situation (Amankwah-Amoah et al., 2021). Even, software development provides flexible and innovative solutions to face other actual crises such as the economic one and climate change (Brem et al., 2021). However, software development projects have reported a lack of success. The-Standish-Group (2015) documented that 60% of software projects are delivered after deadline, 56% are out of budget, 44% are unsatisfactory and 41% do not add value. In addition, 34% do not meet the original objectives when organizational agility is low (KPMG, 2017). Also, Jørgensen (2016a) found that software projects are unsuccessful mainly in time control with a rate of 30%, followed by work efficiency with 24% and budget control with 22%.

From the above we can see that software projects register the biggest failures in delivery times. Hence, Effort Estimation (EE) becomes essential for planning software projects being a key factor for their success (Ibraigheeth and Fadzli, 2019; Schweighofer et al., 2016; Arias et al., 2012). Additionally, the application of agile practices has an important contribution (Mohagheghi and Jørgensen, 2017). In

this regard, Jørgensen (2016a) found that agile elements rise the success rate of agile software projects up to 45% in dimensions such as client benefit, functionality, technical quality, work efficiency, budget control, and delivery on time.

Nevertheless, EE process is complex in agile environments due to a large number of interrelated factors that are constantly evolving (Popli and Chauhan, 2013). Here, the human factor is crucial to get accurate EE to the success of software projects (Mohagheghi and Jørgensen, 2017; Schweighofer et al., 2016; Popli and Chauhan, 2014). So, experience and collaboration between the parties involved influence on the EE process' results. For example, in *Scrum*, the *scrum master* depends on the *development team* to get precise EE to deliver on time and the *development team*, in turn, depends on the support of the *scrum master* to accomplish the sprint goals. Then, there is a mutual dependence relationship, through which the parties pretend to achieve their objectives. However, the *scrum master* does not always have the necessary experience and/or full-time dedication to lead a project (Diebold et al., 2015). In addition, developers could be working in several projects at the same time with different incentives (Jaspan and Sadowski, 2019); also, they engage in behaviours such as *decision hijacking*, *free-raider* and *gold plating* (Moe, 2013; Moe et al., 2012; Gulliksen Stray et al., 2011; Moe et al., 2010; Conoscenti et al., 2019). Therefore, team members could skew their decisions based on their personal or professional priorities producing under-estimations or over-estimations. Of course, these issues impact decision-making during the EE process in Agile Software Development (ASD).

The literature provides a myriad of publications about EE techniques and methodologies, but the impact of mutual dependence relationships on EE in a *Scrum* context have not addressed enough. Notwithstanding that *Scrum* is the most widely used framework (Digital.ai., 2020; Mutiullah et al., 2018; Fustik, 2017; Usman et al., 2015). Thus, managers require support mechanisms to strategically analyze EE within the context of agile software project development, considering contextual factors and the *scrum master* profile.

Agency theory provides a perspective to study mutual dependence relationships between someone called the *principal* and another named the *agent*. Between them, it may be that there is complete or incomplete information. That is, the parties may aware of the skills and experience of each other or not. The former case is likely to occur with insourced mature teams; and the latter happens in global

development environments with offshoring teams as they cannot be directly recruited or observed by the *scrum master*. In this thesis, we model both principal-agent cases through the lens of Game Theory to analyze the impact of the decisions of the parties involved in estimating effort.

1.2 Research objectives

This thesis addresses the research question: ¿How can the best EE strategy be selected considering context factors in ASD projects, as well as the project manager profile?. The following objectives have been set to answer the previous question.

General objective

Model the strategic analysis that supports decision-making for the selection of the best EE strategy considering the context in ASD for the delivery on time of the client's requirements.

Specific objectives

- Identify the strengths and weaknesses of the top-down and bottom-up strategies for EE through a literary review.
- Parameterize the profile of an agile project manager or *scrum master*, based on the profile variants reported in the literature.
- Propose the contextual factors that impact ASD projects from a literary review of ASD projects.
- Propose the different strategic scenarios that can be applied for EE in ASD projects.
- Model a mechanism for the selection of the best EE strategic scenario that considers context factors, as well as the profile of the project manager.
- Validate the research hypotheses

    1. A strategic analysis of the EE process in agile software development projects that integrates context factors allows selecting the best EE strategy to meet the customer requirements on time.
    2. The *scrum master* profile is decisive for the selection of the best EE strategy.

1.3 Methodology

This research follows a quantitative approach where the object of study is the EE process under the context of ASD. A factorial-experimental design (Hernández Sampieri et al., 2014) was performed given that the effect of contextual factors and the *scrum master* profile on EE was strategically analyzed. The following three methodological phases were conducted, the first was *descriptive*, then the second was *predictive* and the last one was *experimental*.

The *descriptive phase* was focused on the characterization of the EE process in the context of ASD projects. The main scientific journals and authors that publish relevant research on software EE were consulted (Sehra et al., 2017). Also, this review was open to the literature on agile development such as the *Agile Processes in Software Engineering and Extreme Programming* volumes. The first three specific objectives were carried out with this information; identification of strengths and weaknesses of top-down and bottom-up EE strategies, parameterization of the *scrum master profile*, and proposition of the contextual factors in ASD projects.

The *predictive phase* was based on the previous results, which revealed fundamental parameters for modeling a strategic decision-making mechanism in EE. This mechanism was designed through formal theories of strategic behaviour, such as the Game Theory. Next, the fourth and fifth objectives were developed. That is, proposition of strategic scenarios, formulation of assumptions, modelling of the EE game integrating contextual factors and the *scrum master* profile. A solution concept for the EE game was proposed, which was tested for each strategic scenario.

Finally, in the *experimental phase*, numerical examples were carried out to validate the hypothesis of this study. The effect of context factors and the *scum master* profile were evaluated in the selection of the EE strategy. This methodology allowed to derive knowledge for ASD that were stated in some propositions and theorems.

1.4 Organization of the thesis

This thesis was structured in the five chapters below:

The current chapter describes the problem statement, the research objectives and the implemented

methodology. The second chapter comprises three major themes of the theoretical framework; It begins by explaining what a strategy is, then concerns related to the strategic analysis of EE and by last addresses EE in ASD settings. The second theme presents some previous studies of game theory in diverse issues of software development, as well as the agency theory perspective in this regard and the impact of moral hazard and teamwork quality in EE. Concerning the third theme, software development projects are addressed from both global and agile approach, delving into the *Scrum* framework. Next, second chapter continues with the EE process and ends with a review of top-down and bottom-up strategies.

The third chapter presents the EE game model for ASD projects, specifically under a *Scrum* framework. The game parameters, the basic assumptions, the players' payoff functions, the solution concept and the Nash equilibrium found are described in detail. In addition, two variations of the game (scenarios) are exposed and compared to the original EE game. Here the equilibria and comparative statics propositions of the EE game are stated, as well as the theorems about the best strategic scenario for each player. Finally, numerical examples show the players' strategies and payoffs by each scenario as the exogenous parameters change.

The fourth chapter introduces the offshore EE game model for AGSD projects. Here, the developer team is working offshore under a *Scrum* framework, while the *scrum master* is located onsite. The game parameters, the basic assumptions, the players' payoff functions, the solution concept and the Bayesian Nash equilibria found are described in detail. Also, the equilibria propositions and theorems of the offshore EE game are stated. Finally, numerical examples show the players' strategies and payoffs as the exogenous parameters change.

The fifth chapter presents the conclusions of the thesis and responds to the research hypotheses. By last, future work is proposed to extend our EE game model.

Chapter 2

Complexity of strategic analysis to estimate effort

2.1 The strategy

The strategy is necessary to set direction, define the organization, focus the effort and promote coordination of activity (Mintzberg, 1987b). In fact, the term strategy comes from the Greek word strategós which means "lead an army" (Carrión Maroto, 2007). In the management area, it is an intended plan, a pattern in a stream of actions, a mediating force, or a collective strategist perspective inside an organization (Mintzberg, 1987a). From the point of view of strategic behaviour, it is a "complete" plan that indicates how a person must act depending on the situation in concern (Von Neumann, 1928). According to Borel's strategy notion, it specifies the deterministic or probabilistic move of an agent facing any possible contingency in a game (Cantala, 2014).

Based on the above, the strategy arises in different ways. Mintzberg (1987b) defines that a strategy is deliberate when intentions are realized as they existed previously, while a strategy is emergent when patterns develop in the absence of intentions or even despite them. Mintzberg also observes the former assumes non-existent external forces and the latter that there may be a total absence of

intentions. It is hardly conceiving completely controllable environments and unintentional individuals. Hence, among these extreme circumstances, a continuum of various strategies develops, in each of which a certain environment, controls and leadership prevail (Mintzberg and Waters, 1985). However, strategy formation is a learning process, whereby intentions change as they are carried out giving rise to emerging strategies, which can get formalized as deliberate ones (Mintzberg, 1978). So, the strategy idea varies according to the circumstances and it is precisely its diversity what makes it adaptable to face complex and unexpected scenarios.

Given the above, it becomes important to identify and assess the influential factors on both the intended strategy and the emerging one. Regarding the first, too many forces collude to hinder strategic decision-making. Nooraie (2012) classifies them by decision-specific, internal organizational, external environmental and management team's characteristics. Concerning the second, the individual vision providing direction can emerge en route and react to unexpected threats and opportunities (Mintzberg and Waters, 1985). Both in the leader and any member of an organization can develop an individual vision. After all, novelties in strategy formation appear at any time, as people are always exploring new ways of doing things (Mintzberg and Lampel, 1999). Thus, it is necessary to understand the routines and roles of all those involved in creating and executing the strategy (Whittington, 1996; Burgelman et al., 2018), and balance vision with participation and rationality to achieve an inclusive strategy (Barbosa Lavarda et al., 2010), that brings benefits for all.

2.2 Strategic analysis of effort estimation

Strategic analysis is the first step for strategy formulation (Papulova and Gazova, 2016), furthermore, it is mandatory for its effective implementation and is defined as "understanding of organization and its environment with respect to long-range perspective" (Vaitkevičius et al., 2006, p. 99),. It is an essential condition to understand the environment inside and outside an entity, and later propose strategic lines of action. Clark (1997) indicates a "situation assessment" before generating, evaluating and selecting strategic alternatives. Thus, comprehensive perspectives range from the study of initial data, organizational position, factors' integration, alternatives proposals, their technical feasibility, evaluation and decision (Acevedo Borrego et al., 2010). Also, there are models that delve into the

evaluation of external and internal forces, the analysis of current strategy, the areas, the groups, as well as examining the impacted roles (Caine and Von Rosing, 2018).

Whatever the approach, a strategic analysis always favors the study of the context and critical factors, opening the view of environment to ensure effective decision-making (Papulova and Gazova, 2016). So, the more examined variables, both quantitative and qualitative, the more broad and long-term the strategic vision will be (Nooraie, 2012; Herrero Martín and Pinedo, 2005); even the possible causes of future changes must be taken into account (Busu, 2018). Here, strategic thinking allows reacting, predicting and anticipating the events into a dynamic environment (Papulova and Gazova, 2016).

2.2.1 Effort and game theory

Game theory contributes to the development of strategic thinking, it is based on studying opponents' behaviour in a tournament. In this, the search for optimal decisions gives a special connotation to the strategy. In 1921, Émile Borel introduced the strategy concept from the point of view of optimal decision analysis of a game's participants (Cantala, 2014). This notion prompted the development of a formal game theory that was established with the publication of Theory of Games and Economic Behavior by **?** in 1994. They describe the zero-sum games concept, extending it to general games with n people and propose the cooperative or formation of coalition behaviour.

Zero-sum games with two players are known as non-cooperative or competitive games, where what one wins is lost by the other. So, the total of players' payoffs equals zero. Here, the parties have completely opposite interests (Feijs, 2001). Reasoning proved by Von Neumann (1928) in Minimax theorem, which states that the function that maximizes the minimum expected gains of a player yields the same result as the function that minimizes the maximum expected losses of his opponent.

Minimax theorem is a particular case the equilibrium point notion introduced by Nash (1951), which is the one at the juncture of a player's best response with his opponent's best response. That is, no player wants to move as there is no other situation that brings them greater benefits. The fundamental idea if that both agents are rational and act simultaneously to maximize their gains (Nash, 1951). Another important contribution of Nash is that cooperative games are a reduction of non-cooperative games (Streb, 2015). In the face of a non-cooperative situation, a prior negotiation process must take

place to arise the cooperation, which steps are added movements of a larger non-cooperative game. Also, Nash proved that non-cooperative finite games have always at least one equilibrium point. This approach was revealed as a general solution in the game theory.

Melvin Dresher and Merrill Flodd (Flood, 1952a,b) studied non-cooperative games based on the Prisoner's Dilemma conception, for which Tucker (1983) later proposed a scheme (see figure 2.1). It is a matrix containing in lines and rows the strategies (confess or not confess) of two people who committed a joint violation of law and face the police separately. Each cell has a pair of values representing the payments for the players (reward or fine), depending on their decisions. For example, the tuple (-200, 100) means that person 1 receives a fine of 200 and person 2 gets a reward of 100. The matrix below shows that person 1's best response will always be to confess as his benefits are maximized regardless of person 2's decision. If person 2 confesses, the fine for person 1 will be -100 instead of -200; whereas if person 2 does not confess, person 1 will get 100. Since it is a symmetrical two-person game, the person 2's best response is to confess as well. Therefore, the conjuncture (Confess, Confess) is an equilibrium point that provides a solution to this problem.

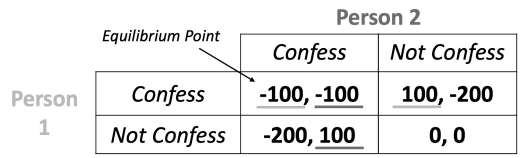

Figure 2.1: The prisoner's dilemma framework.

Source: Own elaboration based on "The Mathematics of Tucker: A Sampler" by A.W. Tucker, 1983, 14(39), p.228. Published by Taylor & Francis.

The Prisoner's Dilemma provides a perspective for analyzing development effort issues; which is "the necessary amount of work to realize a certain quantity of functional effort" (Raith et al., 2013, p. 68), or functionalities of a software requirement. For example, software testing involves opposite interests between programmers and testers; the former pretends to deliver a correct code while the latter is always looking for errors. Feijs (2001) points out the existence of a reward-effort relationship where incentives are necessary to deliver software of a certain quality; Feijs, found that there is a unique equilibrium point here that is not optimal for both players, as happen in the Prisoner's Dilemma.

Kitawa et al. (2016) modeled the code review problem by multiple reviewers based on the snowdrift game, a variant of the Prisoner's Dilemma. They discovered that source code changes are reviewed by only one reviewer, and some reviewers did not continue the review discussion, thereby affecting software quality. Meanwhile, Gavidia-Calderon et al. (2020) propose that adding a single code reviewer to catch 25% of kludges is a cost-effective process intervention that drives developers to write careful code.

Certainly, delivering quality software concerns developers but they fear missing the established deadlines, so they take shortcuts. For their part, managers try to avoid this risk by adding slack time. Interestingly, Austin (2001) found that shortcuts can be eliminated by setting aggressive deadlines, given that a superior pareto equilibria is achieved (High effort, High effort) when deadlines are unattainable. Here, a coordination game emerges between team members when they communicate. In addition, trust between team members is essential for them to cooperate. In this regard, Hazzan and Dubinsky (2005) states that eXtreme Programming (XP) principles improve trust among team members, arising equilibrium points of the (Cooperate, Cooperate) type.

Of course, mutual dependence relationships take place during software development processes; Kautz and Nielsen (2006) examine the relationship between the parties involved by analyzing the impacts on their behaviour derived from the pricing structure. They found that a scenario (Win, Win) is feasible when the parties are aware of their mutual dependence and share goals. In particular, the developers-leader relationship was modeled by Bacon et al. (2010) where leaders should be incentivized to predict accurate estimates, while developers should be incentivized to strive to produce quickly. They explain that the developer's fixed strategy dominates all others leaving to the leader a unique best-response, so this relationship has a unique Nash equilibrium. Finally, Karabiyik et al. (2020) explored the interaction between a scum master and team members by considering only two strategies (active and passive). Although they find equilibrium strategies in the presence of interpersonal relationships and common processes' costs, their study does not establish what is the required effort to build a sprint backlog.

The above investigations show the game theory potential to analyze and understand the diverse relationships in software processes from technical to managerial levels. Here, interactions exist both

vertically and horizontally; through the different organizational levels and among personnel of same areas and groups. Then, competition or collaboration situations may arise and impact software projects results. By this reason, Gavidia-Calderon et al. (2020) propose to apply it to model software process in favor of reducing technical debt or quality deficit.

However, all of these approaches ignore the effort estimation process inside a Scrum context, the most widely used framework for agile software development (Digital.ai., 2020; Mutiullah et al., 2018; Fustik, 2017; Usman et al., 2015). Its strategic analysis can generate knowledge to support decision-making for achieving an effective planning of software projects. It is worth mentioning that Scrum is a framework for agile project management (Azanha et al., 2017), as well as being a software development methodology. Within it, the human factor is crucial to improve effort estimations (Schweighofer et al., 2016; Popli and Chauhan, 2014, 2013). Therefore, the identification of roles involved, the analysis of their relationships and how all of this impact in the effort estimation process are important issues for software projects planning.

2.2.2 The agency theory in software development

Conflicts in software project development can be modeled through the Agency Theory (Hazzan and Dubinsky, 2005; Austin, 2001; Feijs, 2001). The principal-agent relationships are present in such projects; here, the principal delegates a work performance to someone called the agent (Eisenhardt, 1989). So, mutual dependence arises between the parties; both to complete the entrusted work in time and in form, and to obtain a desired benefit. People and their actions are key factors in project outcomes (McLeod and MacDonell, 2011), being essential addressing all difficulties that go out. In software project context, the main role is played by the manager or leader, while the agents are the people who carry out development activities such as programming, analysis, designing or testing. Hereafter, the principal will be referred to as the "project manager" and the agent as the "developer".

Now, agency problems come from the actuation of imperfect agents, being basically two: moral hazard and adverse selection (Fayezi et al., 2019). Moral hazard arises when someone acts in ways that cannot observed and neither contracted directly; meanwhile, adverse selection is the situation where actions can be observed but not verified if they are correct (Holmstrom, 1982). Then, the parties have

limited and different information, that is they have imperfect information. In software projects, there is "incomplete information" when it is impossible to thoroughly observe the developer performance; meanwhile, there is "asymmetric information" when one party has technical and specialized knowledge that the other does not.

The above cases may cause anomalies such as abuse of trust and opportunism. According to Wachnik (2015), moral hazard results from the opportunism and a high level of information asymmetry that, in turn, can lead to a complete failure of projects and increase the transaction costs due to the loss of mutual trust. All of his severely affects software projects, placing them in budget failure positions as well as having dissatisfied customers due to a quality deficit product created by an incomplete work (Ciolkowski et al., 2017). In such scenarios, the question for the project manager is how to reduce moral hazard in order to the developer produce the desired results, that is, he cooperates.

The agency theory addresses cooperative effort issues in the presence of imperfect information; indicating as remedial alternatives to get information about agent behaviour or hire by outcomes. In outcomes-based contracts what matters is the final result and not how it is reached. In contrast, behaviour-based contracts are highly dependent on monitoring systems for getting information about the agent performance. Here, various monitoring strategies are applied such as supervision or pair monitoring systems (Corgnet et al., 2015). Whatever additional information helps improve agency relationships and can be used in developing behaviour-based contracts. Raghu et al. (2004) highlight the importance of setting incentives based on observed performance rather than on outcomes as information structures improve. The question is to what extent the developer should be encouraged based on his willingness to share project risks with the project manager (Lichtenstein, 2004); also, it should be analyzed which is the best strategy to monitoring performance depending on project type and the risks associated.

After all, agency problems arise because results are affected by a certain degree of uncertainty; and when it is high, monitoring is essential (Holmstrom, 1982). In software development, agile methodologies were designed to deal with highly complex and unpredictable environments (Jørgensen, 2019; Marçal et al., 2008); for which they use certain strategies to reduce project risks. Fayezi et al. (2019) state that agility improves through behaviour-based mechanisms since they mitigate agency conflicts. However, the agency theory has not been used in agility research Fayezi et al. (2019); so,

this is an open opportunity to analyze software processes in highly uncertain environments.

2.2.3 Moral hazard in effort estimation

Effort estimation (EE) is central for planning software projects being a success factor for them (Ibraigheeth and Fadzli, 2019; Mohagheghi and Jørgensen, 2017; Arias et al., 2012). An accurate EE contributes to overcome failures that arise during software delivery which exceed deadlines ranging from 30% to 60% (The-Standish-Group, 2015; Jørgensen, 2016a; KPMG, 2017). Hence, EE issues must be identified and strategically analyzed.

In an EE process, developers must agree on the amount of work required to complete all tasks involved in creating a software product. Particularly, in Scrum, leaders prioritize customer requirements and then team members calculate the number of iterations or development cycles according to their joint production capacity. At the beginning of each cycle, a part or increment of the product is planned by breaking down its requirements into individual tasks for which the team estimates the effort required (Zahraoui and Janati Idrissi, 2015). However, unrealistic EE by moral hazard anomalies cause unfinished requirements in a cycle, so they are assigned to the next iteration. Of course, this situation impacts the overall project results.

From an agile perspective, EE is a teamwork (Radu, 2019) where agency issues may emerge as mutual dependence relationships take place. In fact, the team has the responsibility and commitment of setting EE in Scrum, so that the leader is dependent on the developers for getting precise estimations. Now, when a project begins, the leader has imperfect information since he has not worked with the developers; that is, there is a short agency relationship (Eisenhardt, 1989). Here, team's real productivity, which it is a required input for the EE process (Zahraoui and Janati Idrissi, 2015), is unknown. Meanwhile, team members are aware of their capacity, skills, technical knowledge, as well as of their performance level variability (Meyer et al., 2019; Ko, 2019; Treude and Figueira Filho, 2019; Jaspan and Sadowski, 2019). Then, the leader is at a disadvantage due to asymmetry and incomplete information, which open space for moral hazard concerns. Although things are discovered and adjusted as iterations progress, asymmetry prevails as long as technical knowledge gaps between leaders and developers persist.

In addition, developers may deviate from each iteration's goals biased by different reasons. As an example, they could be working in several projects at the same time with different incentives by each project (Jaspan and Sadowski, 2019). Also, some studies have found that developers engage in behaviours such as decision hijacking and free-raider while sprint planning meetings are running (Moe, 2013; Moe et al., 2012; Gulliksen Stray et al., 2011; Moe et al., 2010); and gold plating when they develop the software (Conoscenti et al., 2019). They pursuit of building a good reputation as developers, striving to build high-quality software. Nevertheless, Moe et al. (2010) explain that they focus on perfecting what they consider primordial by their alone, following their own plan without discuss it with their partners. So, team members could skew their decisions based on their personal or professional priorities.

2.2.4 The teamwork quality impact

Team members must be careful when estimating effort, especially with tasks dependencies. If a developer leaves without giving details, the others must re-do the work falling into under-estimations; on the contrary, if the whole team works efficiently, there could be over-estimations (Conoscenti et al., 2019). Therefore, Radu (2019) integrates the Teamwork Quality (TWQ) construct in his EE model to help agile teams to obtain better estimations. Hoegl and Gemuenden (2001) describe TWQ as a measure of collaboration in teams; their TWQ model includes six facets: communication, coordination, balance of member contributions, mutual support, effort and cohesion. Later, Weimar et al. (2017) extended this model by adding trust, value diversity and cooperation factors to explain team performance.

Although, there is a strong positive effect of TWQ on team members' learning and work satisfaction (Lindsjorn et al., 2016), there are also certain forces that hinder an effective teamwork in Scrum environments. Moe et al. (2010) found barriers such as issues with team orientation, team leadership, coordination, as well as highly specialized skills and the division of work. Moreover, agile teams tend to deviate from Scrum framework recommendations, falling into anti-patterns such as progress invisibility and non-existent feedback loops (Eloranta et al., 2016). In this regard, Moe et al. (2010) expose that Scrum has mechanisms designed to ensure good results but they are not easy to carry

out. For example, retrospective and review meetings are considered a waste of time (Moe et al., 2009), so they are not held or are not used to detect problems. As a consequence, both the leader loses the opportunity to obtain information about the team's work, and the team cannot be helped in eliminating the conflicts that impede it go ahead.

2.3 Effort estimation in agile software development

This section explains what a software development project is and how the Effort Estimation (EE) is carried out in them, detailing the approaches and techniques that are used. The agile and global perspective are addressed, specifically under the Scrum framework.

2.3.1 Software projects

Projects represent efforts to implement changes in business firms (Lundin and Söderholm, 1995) for improving something or meeting a need. They are temporary endeavors with a deadline, from which the delivery of a product, service or benefit is expected (Cunha et al., 2016). Thus, its temporary and unique nature stands out, where the effort must be organized through a solid plan for its success (Braude & Bernstein, 2011).

Particularly, a software project comprises the activities required to produce software such as analysis, design, coding, testing and maintenance. For its management, a variety of activities and work structures are incorporated creating paradigms to produce software (Braude and Bernstein, 2011). Below are the different configurations in terms of physical distribution of those who collaborate in software development. Then, the agile development paradigm is approached, whose practices are among the main success factors of a software project (Mohagheghi and Jørgensen, 2017).

2.3.2 Global software development

The Global Software Development (GSD) is also known as globally distributed software development. It means that the team is distributed in diverse geographic locations that even are in various countries, as well as their members may belong to different companies working all together on a specific project (Yadav, 2016). So that, this configures diverse scenarios related with the sourcing strategies of

members in a GSD context. Hence, some terms are used for each scheme; Šmite et al. (2014) defined two main dimensions to classify them, the geographic location and the legal entity. The former accounts for the location of the receiving company, the latter refers to which business unit belong team members. Geographically, onshoring means that the entire team is within the same country, while offshoring come about when team members in another country or even dispersed in different locations. Legally, insourcing means that all developers are internal, while outsourcing is when they are hired by a third party.

Of course, a firm may have departments or branch offices in several countries resulting cases of offshoring with insourcing. Also, a group of firms established in a certain nation could be collaborating on the same project, so it is a scheme of onshoring with outsourcing. In such a way that the source of supply of team members makes up diverse ways of working to produce software. (see figure 2.2).

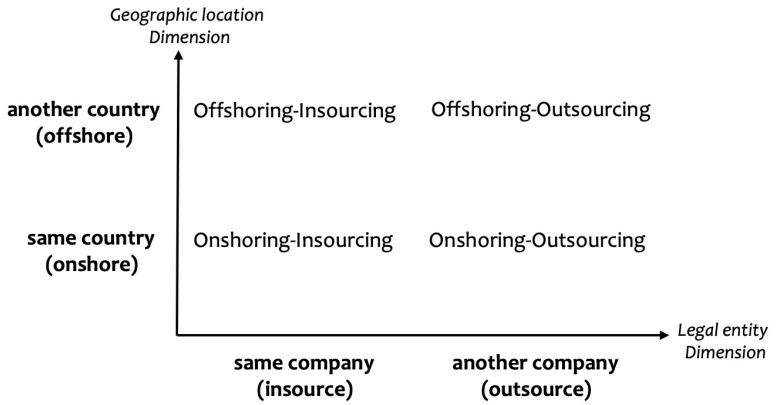

Figure 2.2: Sourcing strategies of team members in GSD.

Source: Elaboration based on "An empirically based terminology and taxonomy for global software engineering" by D. Šmite, C. Wohlin, Z. Galviņa & R. Prikladnicki, 2014, 19, p. 124. Published by © Springer Science+Business Media.

GSD has attractive benefits for software business; it provides access to international talents, large skilled labor pool, closer proximity to new markets, as well as brings innovation opportunities (Wickramaarachchi and Lai, 2017). Of course, the particular conditions of physical dispersion of team members add challenges related to communication, coordination and control (Yadav, 2016). Here, distance means not only physical location, also it is temporal and socio-cultural (Šmite et al., 2014). Regarding to

communication overload, Britto et al. (2015) report that it is bigger in globally distributed projects and affects the accuracy of EE leading to under-estimations. Hence, the EE process is impacted by work dependencies in GSD environments. Wickramaarachchi and Lai (2017) point out that a systematic work allocation method is necessary to get an accurate EE in GSD. In contribution to this, Britto (2015) extended the taxonomy of Šmite et al. (2014), by adding dimensions such as the ways of work and the roles of the participants. For the former, he proposes the centralized, distributed o semi-distributed schemes; and for the latter, the estimators or only data providers roles.

2.3.3 Agile software development

The Agile Software Development (ASD) is a paradigm that address the principles stated in the Agile Manifesto, which is motivated for "uncovering better ways of developing software". The agile way values individuals and interactions, working software, customer collaboration and responding to change (Manifesto, 2001). Agile methodologies were designed to deal with highly complex projects for which it has shown to produce better performance than other paradigms due to its flexible scope, frequent deliveries and the high degree of realization of requirements changes (Jørgensen, 2019). However, the agile paradigm requires to apply it correctly to get their benefits (Moe et al., 2010). The following section describes the Scrum methodology which it is the most widely used framework for agile software development (Digital.ai., 2020; Mutiullah et al., 2018; Fustik, 2017; Usman et al., 2015).

The Scrum framework

Scrum is an approach for planning and managing software projects giving authority to the people of the operative level (Moe et al., 2010). It is a framework where the development team is empowered to define and execute the tasks necessary for successful software development (Braude and Bernstein, 2011). It carries out an interactive process based on transparency, inspection and adaptation pillars (Azanha et al., 2017). The transparency refers to all relevant aspect are visible to all. The inspection is through diverse mechanisms to detect problems hindering the results. The adaptation comes with the adjustments made to reduce poor results. The figure 2.3 shows the artifacts, roles and meetings involved in this framework.

There are basically two types of artifacts, the product backlog and the sprint backlog. The product backlog consists of a list of product requirements described through user stories; whereas a sprint backlog is a subset of tasks selected from the product backlog, with their respective effort estimates (Bruegge and Dutoit, 2010).

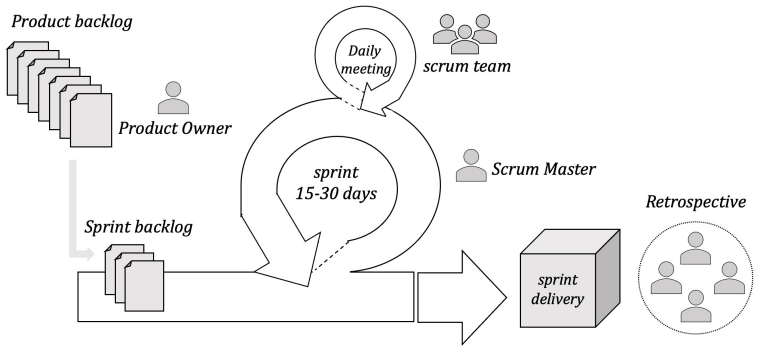

Figure 2.3: The framework Scrum.

Source: Elaboration based on "Desarrollo ágil" por Roger S. Pressman, 2010, Ingeniería del Software Un enfoque práctico, 7, p.70. Copyright © 2010, 2005, 2002 regarding the third edition in Spanish by McGraw-Hill Interamericana Editors.

In this approach, scrum teams are made up of no more than 6 to 9 members (Braude and Bernstein, 2011) with diverse specialties such as analysts, designers, programmers, and testers. They actively participate in planning meetings focuses on the continuous delivery of a product that must be defined, designed, coded and tested in short cycles of one to two weeks or sprints. The sprint constitutes the planning unit where work is estimated (Sommerville, 2011). Here, decision-making is shared by the scrum team having certain challenges such as some developers did not argue with others, lack of team orientation, decision-hijacking and unrealistic plans (Moe et al., 2012). In addition to the team, all the stakeholders involved hold some meetings to face any problem derived from these challenges. So that, the Scrum framework also comprise the following roles and meetings.

This methodology involves also the product owner and the scrum master roles. The former represents the customer and prioritizes user stories that describe product requirements (Bruegge and Dutoit, 2010). The latter's responsibility is to eliminate any obstacle or impediment that prevents the advance of the sprint (Moe et al., 2012). Also, the scrum master conducts the daily meetings, measures progress

and makes decisions (Bruegge and Dutoit, 2010).

Thus, product planning, sprint planning, daily meetings and retrospective meetings are held. In the product planning meeting, requirements are selected and prioritized. Then, each sprint begins with a short planning meeting where the selected requirements are broken down into tasks for which the effort is estimated. Concerning the daily meetings, they last no more than 15 minutes to review the progress, identify problems and plan work for the next day (Pressman, 2010). The retrospective takes places at the end of each sprint with the participation of all the parties involved to reflect about the work done.

2.3.4 The effort estimation process

Under Scrum, planning is a constant activity where effort estimations are crucial to lead to the successful project's completion (Schweighofer et al., 2016). The Effort Estimation (EE) is the process by which the effort required to complete each task is calculated. The effort is the product of the size of what will be built by the productivity level of the development team; it is expressed in terms of man-days or man-months (Sommerville, 2011). However, productivity is discovered as work progresses. Zahraoui and Janati Idrissi (2015) explain that EE in Scrum comprise the next steps. Firstly, comparison of user stories with each other to establish their relative size. Secondly, conversion of size estimates to effort estimates using the assumed team productivity. Finally, re-estimation of effort estimates using the exposed actual productivity, after two or three iterations.

From the above, it can be notice that EE is inherently inaccurate due to the lack of relevant information and subjective measurements (Dragicevic et al., 2017). In fact, at the beginning of a project, information is often sparse and little detailed, resulting in estimates based on incomplete, imperfect knowledge and assumptions (Galorath, 2006). Hence, it is necessary to analyze what is the best strategy for an effective EE process and thus obtain more realistic estimates. Basically, there are two strategies to estimate the effort in ASD projects, the Top-Down and the Bottom-Up (Trendowicz and Jeffery, 2014). The former use mathematical formulations or algorithms to determine effort, and the latter is based on the team experience.

2.3.5 The Top-Down strategy

A Top-Down strategy estimates from an outside view of a project and may use historic information of similar developments (Moløkken-Østvold and Jørgensen, 2005). Here, mathematical formulations are applied to determine the effort, requiring data such as application type, size, programming language and team knowledge. Indeed, this approach employs parametric and machine learning techniques.

Regarding the parametric models, the Constructive Cost Model (COCOMO) was designed to estimate project costs. The first version of COCOMO was published in the eighties (Boehm, 1981), after which several versions were derived to achieve a better fit function. The version COCOMO II is a data-based early effort estimation model (Sommerville, 2011); that is, it is used in the early stages of development. Here, costs and size do not have a linear relationship, as complexity increase costs exponentially. The mathematical formulation is $effort = A * size^B * M$ (Boehm, 2000), where A is a constant determined by local practices and software type, B is the product complexity, M is a measure of project and process attributes and size is the number of code lines (LOC).

However, as software needs grew and evolved, software development became more complex and LOC parameter is no longer considered realistic. Hence, emerge the Function Points (FP) presented by Alan Albrecht in 1978 (Dekkers, 2017). They are a measure of functional size independent of the programming language or technology. It exists diverse standards managed by the International Function Point Users Group (IFPUG). The International Organization for Standardization (ISO) recognizes the standards developed in different places, such as FiSMA -Finnish Software Metrics Association-, NESMA -Nehterlands Function Point Users Group-, UKSMA -United Kingdom Software Metrics Association-, DASMA -Deutsche Anwendergruppe für Software Metrik und Aufwandschätzung-, JFPUG -Japan Function Point Users Group-, PoSMA -Polish Software Metrics Association- and COSMIC -Common Software Measurement International Consortium- (Czarnacka-Chrobot, 2012). Concerning, IFPUG and COSMIC use transactional data related to processes of inputs, outputs, reading and writing (Commeyne et al., 2016). Thus, functional measurement involves identifying as well as classifying elementary processes, files and queries, weighting data and transaction functions, and many more activities (Lavazza and Meli, 2014). Hence, it is time consuming, expensive and also, demands a high level of knowledge (Meli, 2011).

An alternative to reduce functional size computing costs is Simple Function Point (SiFP), which has been proven to be reliable to be used in place of IFPUG (Lavazza and Meli, 2014). Nevertheless, SiFP and IFPUG do not improve EE accuracy in agile methodologies like Scrum (Lenarduzzi et al., 2015). Moreover, practitioners consider that functional methods reduce agility and teams prefer maintain freedom in decision-making during software development (Hacaloglu and Demirors, 2018).

On the subject of Machine Learning based models, Wen et al. (2012) identified the Case Base Reasoning, Neural Networks, Decision Trees, Bayesian Networks and Support Vector Machines as the most widely used. The Case Base Reasoning can deal with poorly understood domains that are also difficult to model, but it has a sensitive similarity function, it does not accept categorical features, nor the absence of values (Wen et al., 2012). With reference to Neural Networks, they tend to over-estimate up to 80% of the time, although they learn non-linear functions despite having noisy data (Wen et al., 2012). As regards to Decision Trees, people without specialized knowledge can visualize a problem clearly and broadly (Satapathy and Rath, 2017); hence, Decision Trees are an intuitive technique that also, works well with categorical features (Wen et al., 2012). In the matter of Bayesian Networks, Dragicevic et al. (2017) proposed them to face problems that generate uncertainty in EE. Additionally, Bayesian Networks combine historical data with expert opinions, learn causal relationships, and perform well with small data sets (Wen et al., 2012), and even incomplete data sets (Zare et al., 2016).

In more complex approaches, Machine Learning techniques are often mixed creating hybrid or assembled models in order to improve EE. Some hybrid examples are morphological neural networks (Araújo et al., 2017), general regression networks (López-Martin, 2011), Bayesian networks optimized through genetic or particle swarm algorithms (Zare et al., 2016), and support vector machines assembled with neural networks (Pospieszny et al., 2018). An assembly consists of several EE techniques that produce a combined result using a rule of either linear or non-linear type. Particularly, the Decision Trees' gear exceeds twice the predictive capacity of Neural Networks' gear (Idri et al., 2016). Even the Random Forest method which couples Decision Trees is better at predicting than a Neural Network working individually (Sharma and Karambir., 2016). Also, Neural Networks assembled with Support Vector Machines produce more accurate results than when they run by their alone (Idri et al., 2016). After

all, assemblies can function as a support tool to predict effort, complementing the EE process results generated from other strategies.

2.3.6 The Bottom-Up strategy

The Bottom-Up strategy concerns an inside view of a project that is divided into tasks, estimated individually and later adding everything to compute the total effort (Moløkken-Østvold and Jørgensen, 2005). So that, the software to develop is broken down into their elemental functions generating a detailed structure of tasks for the project's general estimation called Work Breakdown Structure (Galorath, 2006). All this, it is based on team experience. However, this approach is harshly criticized in ASD environments for being subjective and relative (Hacaloglu and Demirors, 2018), sensitive to experience and prone to bias (Tanveer et al., 2017) due to customer expectations, irrelevant and misleading information, estimation request format, and the unitosity effect (Jørgensen, 2016b; Jørgensen and Grimstad, 2011; Jørgensen and Halkjelsvik, 2010; Jørgensen and Sjøberg, 2004). Furthermore, experience-based estimates could be inconsistent as much as 71% and differ even when a same person estimates the same task at different occasions (Grimstad and Jørgensen, 2007).

After all, expert-based estimation method is valued for its benefits. **?**, Lenarduzzi et al. (2015) argue that in some situations these EEs are more accurate than those obtained by means of formal models. In any case, models always depend on expert judgments to produce their inputs related with complexity, team skill and customer's requirements (Jørgensen and Boehm, 2009). Indeed, (**?**) found that EE are more accurate when experts possess and efficiently apply domain knowledge that models do not include, when uncertainty is low, and when the heuristic used are valid. Moreover, formal models do not consider leader-programmer conflicts and their impact on project's productivity (Jørgensen and Boehm, 2009).

Now, among the techniques used under this approach are User Stories, Planning Poker, Analogies and Use Cases. The User Stories consist of short descriptions of each software requirement written in the customer's language (Mahnic and Hovelja, 2012) and their details are discovered during the collaboration at a sprint (Torrecilla-Salinas et al., 2015). They are estimated by using the Planning Poker technique; which is based on team members' agreement on the list of requirements or product

backlog. It uses a set of cards numbered with a non-linear series, such as Fibonacci, to estimate the relative size of each requirement. The numbers are named as story points and indicate how hard the story is (Sharma and Karambir., 2016). Planning Poker ensures that all developers participate in the EE process and feel ownership of the estimates, thus assuming more responsibility for them (Haugen, 2006).

Regarding Analogies, they are based on a Case Base Reasoning. This technique employs the k nearest neighbor algorithm for selecting the most similar case to achieve more accurate prediction of effort. It follows the hypothesis that software projects with similar features should require similar effort (Phannachitta et al., 2017). So, it retrieves the most similar past project cases from a data base, then apply a solution adaptation by which refine and solve the EE most suitable to the new case. However, inconsistent behaviours in solution adaptation techniques manifest its instability problems where the right feature selection method influences the accuracy performance (Phannachitta et al., 2017). By last, the Use Cases method can be applied in the early stages of development. As they are concrete specifications for each case, Huanca and Oré (2016) found that standardizing them and then correctly accounting their transactions are key points to refine EE and this should reduce inaccuracies. Meanwhile, Azzeh and Nassif (2017) studied the relationship between project productivity and environmental factors in this method. They recommend not taking the latter into account when calculating Use Cases points and using them only for predicting productivity; this allows to estimate early effort very well.

Chapter 3

EE in agile software development with insourcing mature teams

3.1 Introduction

Software development represents an opportunity to provide flexible and non-material solutions for carrying out activities from almost all social and economic sectors. Among its advantages, software development (SD) facilitates communication and allows tasks' distribution in different places (Brem et al., 2021; Yu et al., 2021; Belhadi et al., 2021; Munien and Telukdarie, 2021). However, its successful implementation depends on effort estimation (EE) due to projects' complexity (Mohagheghi and Jørgensen, 2017; Arias et al., 2012). Even more, software development involves agents with different objectives (Popli and Chauhan, 2014) which makes effort estimation (EE) a big challenge to overcome the failures that arise during software delivery (Jørgensen, 2016a; KPMG, 2017; The-Standish-Group, 2015).

Concerning Scrum, EE is essential for planning each development cycle or sprint (Azanha et al., 2017; Schweighofer et al., 2016). Thus, such estimation lies in the relationship of mutual dependence between the project leader (scrum master) and the development team to accomplish the project's

objectives. Moreover, the project's completion depends on how the leader-team relationship unfolds (Grapenthin et al., 2014). While the scrum master aims to produce the highest business value in each sprint, the team wants to maximize its profits by exerting a minimum effort. In this regard, some studies have found that developers tend to underestimate effort by 25% or higher even (Prieto-Bustamante, 2020; Usman et al., 2015). In this research, we analyze effort estimation as the result of a strategic interaction between the scrum master (leader) and a mature development team (follower) during insourcing software development. Particularly, we focus on projects that rely on mature teams since antiquity does not guarantee on-time software delivery (Lundene and Mohagheghi, 2018; Usman et al., 2018). As the scrum master is in charge of providing resources to the team, we investigate the magnitude of resources provided depending on the scrum master type and the impact of the team type on effort levels. In addition, we analyze the agents' strategies at equilibrium under three different scenarios that outline the sequence of interactions that occur around scrum meetings.

Our modelling approach is based on a principal-agent conflict since software development is a sequential interaction under which the scrum master (principal) guides the mature team (agent) to develop software (Eisenhardt, 1989). Since we focus on insourcing projects with mature teams, we model the previous interaction as a sequential game of complete information (Spiegler et al., 2020; Srivastava and Jain, 2017; Hoda et al., 2013). As usual, we consider that the scrum master provides resources to the development team while the team exerts effort to develop a software's project.

Effort estimation is crucial in mature teams, given that many firms prefer insourcing development instead of outsourcing (Naik, 2016; Omar et al., 2016; Paramanantham et al., 2019). Specifically, insourcing provides firms with significant decision-making control (Chudzicka, 2013; Moe et al., 2014). Our work is closely related to those investigations that analyze the mutual dependence relationships during software development. For example, software testing involves the interaction between programmers and testers; in this case, Feijs (2001) points out the necessity of incentives to develop projects with a minimum quality level. Also, Kitawa et al. (2016) show that a code reviewer chooses a different action from his co-reviewer depending on how large his benefit-cost ratio is. Gavidia-Calderon et al. (2020) propose that adding a single code reviewer to catch 25% of kludges is a cost-effective process intervention that drives developers to write careful code. Concerning accomplishing deadlines for

product delivery, Austin (2001) states that aggressive deadlines can eliminate shortcuts taken by programmers since software development cast similarities with a coordination game. Even more, Hazzan and Dubinsky (2005) find that trust leads to cooperative solutions.

Regarding the mutual dependence between the project leader and developers, Bacon et al. (2010) developed a game-theoretical model where leaders estimate effort while developers strive on production. Karabiyik et al. (2020) explore the interaction between a scrum master and team members by considering only two strategies (active and passive). Although they find equilibrium strategies in the presence of interpersonal relationships and common processes' costs, their study does not establish what is the required effort to build a sprint backlog. In contrast, our main results provide closed-form solutions concerning effort estimation at equilibrium; this is our major contribution.

In general, the previous approaches ignore the effort estimation process inside a Scrum context, despite being the most widely used framework for agile software development (Digital.ai., 2020; Mutiullah et al., 2018; Fustik, 2017; Usman et al., 2015). As far as our knowledge, we are the first ones that analyze effort estimation in a Scrum context through the lens of the Game Theory. Even more, we provide a closed-form solution to compute EE in terms of parameters related to agents, environment, and sprint backlog characteristics. Our findings can support scrum leaders' decisions in companies where insourcing practices are central to their competitive strategy.

This chapter is organized into seven sections. The second section explains how a software project development is planned under the Scrum context; hence, this section provides general concepts concerning Scrum master. Later, in the third section, we present the game-theoretical model that establishes effort in Scrum projects; also, we expose the solution for our game. Next, we describe the payoff functions and compute the sub-game perfect Nash equilibria in the fourth section. Section five focuses on analyzing variations of our base game; we compute their sub-game perfect Nash equilibria to compare agents' behaviour at different scenarios. The sixth section shows some numerical examples. Finally, we discuss the conclusions and future work.

3.2 Product planning in Scrum

Scrum is a SD methodology under which projects are carried out by sprints, which we refer to as iterations. Hence, customers partially receive their projects through a planning process that establishes the number of deliveries and their lasting time. So, the scrum master gets from the product owner the *product backlog*, a prioritized list with $k \in \mathbb{N}$ user stories that describe the customer's requirements that the team must develop (see Block A in Figure 3.1).

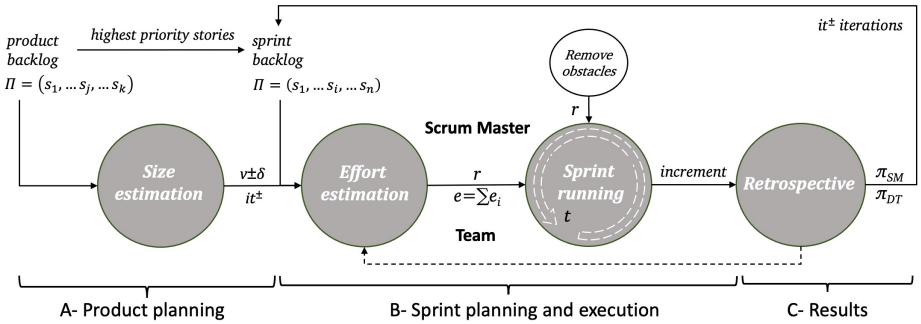

Figure 3.1: The sequence of events around the EE game under the Scrum context.

Source: Own elaboration.

Later, the development team first addresses the user's story with the highest priority in the *product backlog*. Then, the team estimates the size of each user story in story points (sp), which is a measurement unit that compares stories in the backlog (Zahraoui and Janati Idrissi, 2015). The story points allow the team to establish its **iteration velocity** v, representing the number of stories the team can develop in a time t. In a consolidated or mature team, the team truthfully communicates its iteration velocity to the scrum master. So, it is appealing to assume the existence of complete information in this interaction.

The previous paragraph describes how Scrum cycles or sprints are time-boxed (Torrecilla-Salinas et al., 2015). However, it is necessary to set a **tolerance range** δ around v due to the complexity of new projects and the fact that resources vary during software development. So, the scrum master and the team exogenously make a preliminary analysis concerning projects' risks and requirements to set δ. Let $it^{-} = k/v + \delta$ be the **minimum number of iterations or partial deliveries** of the total

product, whereas $it^{+} = k/v - \delta$ is the **maximum number of iterations**.

Given the parameters v, δ, and k, it is possible to establish deadlines and delivery limits that contribute to product planning. Also, the priority list contributes to split the *product backlog* into several *sprint backlogs*, which will be built through iterations. Within each iteration, the team estimates effort e, while the scrum master provides resources r to drive agile development progress. Finally, agents receive their benefit (π_{SM}, π_{DT}) for participating in the sprint.

3.3 The model

In this research, we focus on analyzing a generic iteration concerning an insourcing SD process. So, we only consider the interaction between the scrum master and the development team during the development of a single sprint.

3.3.1 Basics

We consider two types of agents: i. the scrum master (SM) and ii. the development team (DT). Given that scrum teams self-organize, DT members share goals and make decisions collectively (Srivastava and Jain, 2017); thus, we assume DT makes decisions as a single player. So, the set of players is $J = \{SM,\ DT\}$.

The DT's decision variable is the effort that they need to perform tasks for building the story points at the current iteration. In other words, the development team chooses the effort that it commits in every sprint; hence, the DT's effort is a vector $\bar{e} = (e_1, e_2,\ ..., e_n)$ where e_i is the effort that DT spends for each story i in the sprint backlog (see block B in Figure 3.1). As it is usual, total **effort estimation** for the iteration is the sum of all stories' effort, i.e., we have that

$$e = \sum_{i=1}^{n} e_i. \tag{3.1}$$

So, the set of all actions of DT is $A_{DT} = \{e \in \mathbb{R} | v - \delta \leq e_i \leq v + \delta \text{ for all } \quad i = 1,\ \ldots, n\}$.

Concerning the SM, we know that SM is in charge of providing an environment of support and motivation for DT. In this sense, the SM is a mediator and facilitator to remove obstacles and conflicts (Villegas Gómez et al., 2016; Srivastava and Jain, 2017). Thus, we consider that the SM

chooses the number of resources r that the DT can use during its participation stage. The set of all SM's actions is $A_{SM} = \{r | r \geq 0\}$.

Software development does not only depend on resources and the interaction among agents in J; its successful completion also rests on agents' behavioural features, which we refer to as types. The DT's type h varies according to team members' knowledge, skills, and experience (Čelar et al., 2014); let $T_{DT} = \{h | 0 \leq h \leq 1\}$ be the set of all DT's types. If type $h = 0$, the DT does not have the knowledge, skills, and experience to cope with the sprint objectives, whereas $h = 1$ means that DT is fully capable of performing all the sprint's tasks.

The SM's type f summarizes the degree of scrum master's flexibility in their role as a leader (Ramos and Vilela Junior, 2017; Sabbagh, 2013; Quinn et al., 1996). Let $T_{SM} = \{f | 0 \leq f \leq 1\}$ be the set of all SM's types. If SM has a type $f = 0$, the SM does not provide any support (resources) due to a narrow profile. On the contrary, a type $f = 1$ states that SM offers all the necessary resources that DT requests; in this case, the SM behaves as a facilitator, negotiator and protector leader (Shastri et al., 2021).

A mature team is characterized by having a high level of communication between DT and SM. This means that all agents in J know behavioural features of each other (Lundene and Mohagheghi, 2018; Hoda et al., 2013). Thus, we consider that the types vector $(f,\ h) \in T_{SM} \times T_{DT}$ is common knowledge.

3.3.2 Game description

It is worth remembering that we keep fixed an iteration to analyze EE in software development. Given the *sprint backlog* $\Pi = (s_1, s_2, ...s_n)$, the interaction between the SM and DT is a two-stage game that unfolds as follows

Stage 1. The SM chooses the additional resources r that she provides to DT for the iteration's development.

Stage 2. The DT observes resources r and establishes the effort e to exert during the iteration.

The above states that SM participates in the sprint planning meeting to provide resources to DT in a unique stage (see Figure 3.2). Then, the resources provided by SM to DT are $K + r$, which means

basic infrastructure plus additional resources to develop the *sprint backlog*.

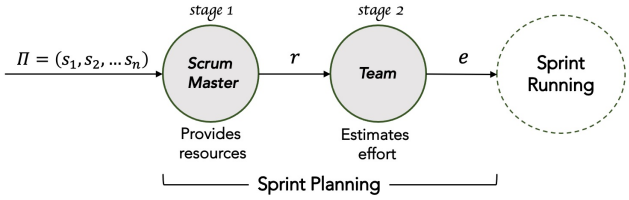

Figure 3.2: Effort estimation game.

Source: Own elaboration.

3.3.3 The solution concept

Before describing the solution concept for the previous game, we introduce some notation to describe players' strategies. Since the scrum master chooses resources in the first stage, her set of strategies S_{SM} is the same as her set of actions, i.e., $S_{SM} = A_{SM}$. In contrast, the DT observes the number of resources provided by SM at the beginning of stage two. Thus, given an amount of resources r, the DT chooses the necessary effort e to complete tasks in the iteration. Consequently, a strategy ϵ of DT is a function from S_{SM} to A_{DT}. Let S_{DT} be the set of all possible strategies of DT. A strategy profile is a vector $(r, \epsilon) \in S_{SM} \times S_{DT}$.

At the end of the game, both players observe the other's strategies and get of payoff. Players' payoffs are functions from $S_{SM} \times S_{DT}$ to $\mathbb{R}$. Let $\pi_{SM}(\cdot)$ and $\pi_{DT}(\cdot)$ be the payoffs function of SM and DT, respectively.

Remembering that SM and DT interact in a sequential game. Hence, we have

Definition 3.1. *The strategies profile (r^*, ϵ^*) is a sub-game perfect Nash equilibrium if:*

(a) The DT's payoff is greater than or equal to other payoffs he would receive for any different effort it can exert, i.e., $\pi_{DT}(r, \epsilon^*(r)) \geq \pi_{DT}(\epsilon(r), r)$ for all $\epsilon \in S_{DT}$.

(b) The SM's payoff is greater or equal to any other payoff he would receive for any different number of resources she can provide, i.e., $\pi_{SM}(r^*, \epsilon^*(r^*)) \geq \pi_{SM}(\epsilon^*(r), r)$ for any strategy $r \in S_{SM}$.

3.4 Equilibrium

We compute the sub-game perfect Nash equilibria by following the backwards induction process. By considering the game described in section 3.3.2, firstly, we calculate the solution of stage 2 in which the DT maximizes its payoffs given a number of resources r. After that, we search for the SM's equilibrium strategy that maximizes her payoffs. Hence, before computing the set of sub-game perfect Nash equilibria, we describe each player's payoff function.

3.4.1 Payoff functions

Note that the SM is in charge of the iteration's execution. So, SM's output depends on DT's effort because the SM is not directly involved in software development; in other words, DT's effort is an input for the SM activities, and SM deals with the costs of providing resources. Below, we describe the production and cost functions.

Concerning costs, SM faces a cost function $c(r)$ which we assume has increasing marginal costs, i.e., we have that $c' > 0$ and $c'' > 0$. It is worth mentioning that SM faces two types of costs. The first ones are related to the basic infrastructure that SD needs, which we denote by K, and the second ones relate to provide a quantity of resources r that DT needs for a specific project. Typically, the software development literature considers a quadratic cost function $a * cost^2$ where a is an adjustment weight that summarizes the SM features and exogenous factors (Lee and Kim, 2013). For example, costs diminish as f increases, i.e., a more flexible SM possesses soft skills that help to reduce DT's costs in agile environments (Ramos and Vilela Junior, 2017). Also, the environment stability τ diminishes costs since it captures the absence of issues related to team members, technology, and process implementation (Ziauddin and Zia, 2012).

Assumption 3.1. *The SM faces the following cost function*

$$c_{SM}(r) = \frac{(K+r)^2}{f\tau}.$$

Note that the SM's production function F_{SM} transforms a basket of inputs (r, e) into the software project. We consider that F has diminishing marginal returns, i.e., ${F_{SM}}' > 0$ and ${F_{SM}}'' < 0$. Moreover, the final production depends on total resources $K + r$ and total effort $G + e$, where G

represents basic activities that DT has to perform. The literature on Software Engineering represents software's production through Cobb-Douglas (Trendowicz and Jeffery, 2014) or sigmoidal (Cuadrado-García et al., 2011) production functions. Specifically, we consider a Cobb-Douglas production function where the output elasticity of resources and effort is 1 and $\frac{1}{2}$, respectively. This means that striving over time is difficult, while increasing resources focused on motivating and eliminating team conflicts boosts produtivity (Desjardins, 2012; Melo et al., 2012; Beecham et al., 2008).

Assumption 3.2. *The SM's production function is*

$$F(r, e) = (K + r)\sqrt{G + e}.$$

Finally, to simplify the analysis, we assume that the price of SM's output is normalized to one. So, by Assumptions 3.1 and 3.2, the scrum master's benefit function is

$$\pi_{SM}(r, e) = (K + r)\sqrt{G + e} - \frac{(K + r)^2}{f\tau}. \tag{3.2}$$

It is worth recalling that SM facilitates DT's job through the provision of resources. In other words, the SM pursues that the DT focuses on fulfilling the sprint goals, which means that DT outputs the software project by transforming its effort and the amount of resources that it gets. As in the SM's case, we assume that DT has Cobb-Douglas production function F_{DT} with increasing marginal returns since additional resources and effort boost the DT's productivity. So, DT benefits from each additional resource that he receives to perform its activities. For example, the development team earns knowledge and skills when SM provides to him specialized training or new technology (Beecham et al., 2008). Hence, we assume the following DT production function.

Assumption 3.3. *The production function of the development team is*

$$F_{DT}(r, e) = (K + r)(G + e).$$

Now, the costs that DT faces consist of fixed expenses G (food, transportation, and certifications). In addition, the effort e that DT exerts induces a variable cost since SD project differ on the complexity of sprint stories, which we represent by using γ (Ziauddin and Zia, 2012). Note that DT's type diminishes the cost of exerting effort since such parameter measures the DT's abilities (Čelar et al., 2014).

Assumption 3.4. *The cost function of DT is*

$$c_{DT}(e) = \gamma \frac{(G+e)^2}{h}.$$

Finally, we assume that DT gets a fixed income I, representing the basic wage for participating in a SD project. By Assumptions 3.3 and 3.4, the benefit function of DT is

$$\pi_{DT}(r,e) = I + (K+r)(G+e) - \gamma \frac{(G+e)^2}{h}. \tag{3.3}$$

Table 3.1 summarizes all the model's parameters.

Table 3.1: Model Parameters

Parameter	Description
e	Estimated effort for the sprint, $v - \delta \leq e \leq v + \delta$.
r	Additional resources required by DT, $r \geq 0$.
I	DT monetary income, $I > 0$.
K	Basic infrastructure (hardware, software), $K > 0$.
G	DT fixed expenses (food, transportation), $G > 0$.
f	Scrum master flexibility, $0 \leq f \leq 1$. (Ramos and Vilela Junior, 2017).
h	Development team experience, $0 \leq h \leq 1$. (Čelar et al., 2014).
γ	Sprint backlog complexity, $0 \leq \gamma \leq 1$. (Ziauddin and Zia, 2012).
τ	Environmental stability, $0.43 \leq \tau \leq 1$ (Ziauddin and Zia, 2012).

3.4.2 DT behaviour at equilibrium

Given that DT and SM interact in a two-stage game, we compute the sub-game perfect Nash equilibrium by following the backward induction process. Hence, we first solve the game's last stage, where DT chooses the effort that it exerts during the SD project given the additional resources r provided by SM. Since DT is rational, it exerts the effort e that maximizes its benefit. Consequently, at stage 2, the DT's decision problem is

$$\max_{e \in \mathbb{R}_+} \pi_{DT}(r,e),$$

which we solve by using the first and second-order conditions.

The first-order condition states that DT's equilibrium strategy is the solution of equation $\partial \pi_{DT}/\partial e = 0$. So, we have that

$$\frac{\partial \pi_{DT}}{\partial e} = K + r - \frac{2\ \gamma\ \ (G+e)}{h}.$$

Hence, the critical points of π_{DT} are the solutions of the equation

$$K + r - \frac{2\ \gamma\ \ (G+e)}{h} = 0.$$

By considering that $\gamma \neq 0$, the critical point of π_{DT} is

$$\epsilon^* = \frac{(K+r)\,h}{2\ \gamma} - G. \tag{3.4}$$

We use the second derivative criterion to verify if expression (3.4) is the DT's equilibrium strategy at stage 2. Note that the second derivative of π_{DT}

$$\frac{\partial^2 \pi_{DT}}{\partial e^2} = -\frac{2\ \gamma}{h},$$

which is always negative. Specifically, we have that $\frac{\partial^2 \pi_{DT}}{\partial e^2}(\epsilon^*) < 0$. Consequently, we conclude that strategy ϵ^* is the strategy that maximizes the DT's benefit.

3.4.3 Resources at stage 1 equilibrium

Moving backwards to the game's first stage, the DT's strategy ϵ^* is introduced in the SM's benefit function π_{SM}, see expression (3.2). Consequently, the scrum master benefit is rewritten as follows

$$\pi_{SM}(r,\epsilon) = (K+r)\sqrt{G + \frac{(K+r)\,h}{2\ \gamma} - G} - \frac{(K+r)^2}{f\tau}.$$

Thus, at stage 1, the SM's decision problem is

$$\max_{r \in A_{SM}} \pi_{SM}(r, \epsilon^*(r)),$$

which we solve by using the first and second-order conditions.

First, we need to solve the equation $\partial\pi_{SM}/\partial r = 0$ to get the critical points of the SM's benefit function. We have that

$$\frac{\partial\pi_{SM}}{\partial r} = \frac{(K+r)\,h}{2\sqrt{2}\gamma\sqrt{\frac{(K+r)\;h}{\gamma}}} + \frac{\sqrt{\frac{(K+r)h}{\gamma}}}{\sqrt{2}} - \frac{2\;(K+r)}{f\tau}.$$

By solving $\partial\pi_{SM}/\partial r = 0$, the critical point of π_{SM} is

$$r^* = \frac{9\tau^2 h f^2}{32\gamma} - K \tag{3.5}$$

Now, we need to determine if the unique critical point r^* is a local minimum or local maximum. To this end, we use the second derivative criterion; the second derivative of π_{SM} is

$$\frac{\partial^2\pi_{SM}}{\partial {r^2}} = -\frac{2}{f\tau} + \frac{3\;\sqrt{\frac{(K+r)h}{\gamma}}}{4\;\sqrt{2}\,(K+r)}.$$

Then, we evaluate r^* into the second derivative of π_{SM}; we get that

$$\frac{\partial^2\pi_{SM}}{\partial {r^2}}(r^*) = -\frac{2}{f\tau} + \frac{3\;\sqrt{\frac{\left(K+\frac{9\;\tau^2\;h\;f^2}{32\;\gamma}-K\right)\;h}{\gamma}}}{4\;\sqrt{2}\;\left(K+\frac{9\;\tau^2\;h\;f^2}{32\;\gamma}-K\right)}.$$

By simplifying the previous expression, we get that

$$\begin{aligned}\frac{\partial^2\pi_{SM}}{\partial {r^2}}(r^*) &= -\frac{2}{f\tau} + \frac{\frac{9\;\tau\;h\;f}{\sqrt{32}\;\gamma}}{\frac{36\;\sqrt{2}\;\tau^2\;h\;f^2}{32\;\gamma}} \\ &= -\frac{2}{f\tau} + \frac{288\;\tau\;h\;f\;\gamma}{36\;\sqrt{2}\;\sqrt{32}\;\tau^2\;h\;f^2\;\gamma} \\ &= -\frac{2}{f\tau} + \frac{288}{288\;\tau f} = -\frac{1}{f\tau}.\end{aligned}$$

The previous expression points out that $\frac{\partial^2\pi_{SM}}{\partial {r^2}}(r^*) < 0$, i.e., the scrum master maximizes her benefit function by following the strategy r^*. Hence, we conclude that resource function in expression (3.5) is the equilibrium strategy for SM.

Finally, we substitute r^* into $\epsilon^*(r^*)$. In other words, the sub-game perfect Nash equilibrium is the profile $(r^*, \epsilon^*(r^*))$. The following proposition formalizes the previous statement.

Proposition 3.1. *Consider that $\gamma > 0$ in the two-stage EE game between SM and DT. There is a unique sub-game perfect Nash equilibrium given by the strategies profile*

$$(r^*, \epsilon^*(r^*)) = \left(\frac{9\;\tau^2 h f^2}{32\;\gamma} - K, \frac{9\;\tau^2 h^2 f^2}{64\;\gamma^2} - G\right).$$

3.5 Interaction's variations

The impact of SM resources provision on DT's effort may vary depending on how they are deployed during their interaction in the Scrum framework. Since the last approach is based on outlining events around the sprint (the planning meeting, daily meetings, and the retrospective), the scrum master and her development team may interact in different sequences. So, SM and DT are free to modify their interventions in the effort estimation game (EE) described in section 3.3.2. The following subsections discuss possible scenarios under which the scrum master and the development team change their roles as leader or follower. So, we discuss how such interaction's structures modify agents' strategies at equilibrium.

3.5.1 Scenario 1

Our first variation considers that DT starts its activities before the SM provides additional resources. Then, the DT behaves as the leader while the SM is the follower that responds to DT's requests for additional resources. Formally, in this case, given the *sprint backlog* $\Pi = (s_1, s_2, ...s_n)$, the interaction between SM and DT proceeds as follows:

Stage 1. The DT establishes the effort e to exert during the sprint.

Stage 2. SM observes the effort e. Then, SM chooses the resources r that she wants to provide to DT during the iteration's development.

Since SM is the follower, scenario 1 describes a situation where SM waits to know the needs of DT to accomplish the sprint's goals. Hence, in addition to the basic infrastructure K, SM focuses on providing additional resources r to remove obstacles during the project's development (see Figure 3.3).

Notice that benefit functions are the same as the benefit functions of the original effort estimation game. However, the backward induction process to compute the game's solution changes. So, we first solve the SM's problem at stage 2. Later, we compute the DT's effort at equilibrium in stage 1.

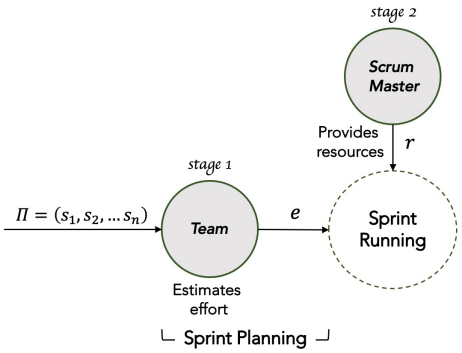

Figure 3.3: Scenario 1.

Source: Own elaboration.

Equilibrium resources at stage 2

We consider that SM is rational, so, she provides the additional resources r that allow her to accomplish the sprint's goals with which she maximizes her benefits. Having that the SM's benefit is

$$\pi_{SM}(r,e) = (K+r)\ \sqrt{G+e}\ - \frac{(K+r)^2}{f\ \tau}.$$

Thus, the SM's decision problem at stage 2 is

$$\max_{r\in\mathbb{R}_+} \pi_{SM}(r,e).$$

As before, we solve the previous problem by using the first and second-order conditions. The first-order condition states that SM's optimal strategies, at stage 2, are the solutions of $\partial\pi_{SM}/\partial r = 0$, where the first derivative of SM's benefits is

$$\frac{\partial \pi_{SM}}{\partial r} = \sqrt{G+e}\ - \frac{2\ (K+r)}{f\tau}.$$

By solving the previous expression equal to zero, the critical point of π_{SM} is

$$r^{*2} = \frac{f\tau\sqrt{G+e}}{2} - K. \tag{3.6}$$

Now, the second derivative is

$$\frac{\partial^2 \pi_{SM}}{\partial r^2} = -\frac{2}{f\tau}.$$

Since $\partial^2\pi_{SM}/\partial r^2 < 0$, expression (3.6) is the equilibrium strategy for SM at scenario 1.

Effort at stage 1 equilibrium

Next, in the backward induction process, we search for the equilibrium strategy of the DT given a quantity of resources r^{*2}. In other words, the DT's decision problem at stage 1 is

$$\max_{e \in A_{DT}} \pi_{DT}(r^{*2}(e), e).$$

Hence, we evaluate the SM's strategy (expression (3.6)) into the DT's payoff function π_{DT}. So, the DT's benefit function is rewritten as follows

$$\pi_{DT}\left(r^{*2}, e\right) = I + \left(K + \frac{f\tau\sqrt{G+e}}{2} - K\right)(G+e) - \gamma\frac{(G+e)^2}{h}.$$

To maximize the benefit of DT, we first compute all the critical points of function $\pi_{DT}\left(r^{*2}, e\right)$. So, we need to solve the equation $\partial\pi_{DT}/\partial e = 0$, where

$$\frac{\partial\pi_{DT}}{\partial e} = \frac{3f\tau\sqrt{(G+e)}}{4} - \frac{2\ \gamma\ \ (G+e)}{h}.$$

By making the necessary algebra, π_{DT} has a unique critical point ϵ^{*2}, which is

$$\epsilon^{*2} = \frac{9h^2f^2\tau^2}{64\ \gamma^2} - G. \tag{3.7}$$

Once again, we need to verify that the previous critical point satisfies the second derivative criterion to determine if ϵ^{*2} is a local minimum or local maximum. The second derivative of π_{DT} is

$$\frac{\partial^2\pi_{DT}}{\partial e^2} = -\frac{2\ \gamma}{h} + \frac{3f\tau}{8\sqrt{G+e}}.$$

Now, we substitute ϵ^{*2} into the second derivative. So, we have that

$$\begin{aligned}\frac{\partial^2\pi_{DT}}{\partial e^2}(\epsilon^{*2}) &= -\frac{2\ \gamma}{h} + \frac{3f\tau}{8\ \sqrt{G + \frac{9h^2f^2\tau^2}{64\gamma^2}} - G} \\ &= -\frac{2\ \gamma}{h} + \frac{3f\tau}{\frac{24\ h\ f\ \tau}{8\ \gamma}}.\end{aligned}$$

By rearranging the numerator and denominator of the previous expression, we can simplify it as follows

$$\frac{\partial^2\pi_{DT}}{\partial e^2}(\epsilon^{*2}) = -\frac{2\gamma}{h} + \frac{24\gamma f\tau}{24hf\tau}.$$

Therefore, we get that

$$\frac{\partial^2 \pi_{DT}}{\partial_e^2}(\epsilon^{*2}) = -\frac{\gamma}{h} < 0,$$

that is to say, the strategy ϵ^{*2} is a local maximum of the DT's benefit function. Hence, we conclude that expression (3.7) is the equilibrium strategy for DT at stage 1. Finally, we substitute ϵ^{*2} into $r^{*2}(\epsilon^{*2})$ to obtain the sub-game perfect Nash equilibrium profile $(r^{*2}(\epsilon^{*2}), \epsilon^{*2})$. The above discussion is summarized in the following proposition.

Proposition 3.2. *Given the interaction between SM and DT in the scenario 1 of the EE game, the sub-game perfect Nash equilibrium is the profile of strategies*

$$(r^{*2}, \epsilon^{*2}) = \left(\frac{3\ hf^2\ \tau^2}{16\ \gamma} - K, \frac{9\ h^2 f^2\ \tau^2}{64\ \gamma^2} - G \right).$$

.

3.5.2 Scenario 2

In a second game's variation, we consider that SM provides resources in two stages: before the sprint starts and while it is running. In this case, given the *sprint backlog* $\Pi = (s_1, s_2, ...s_n)$, the game between SM and DT is now a three-stage game that unfolds as follows:

Stage 1. SM chooses the additional resources r_1 that she wants to provide for the iteration's development.

Stage 2. The DT observes resources r_1 and establishes the effort e to develop the sprint backlog.

Stage 3. SM observes the effort e. Then, SM chooses the additional resources r_2 that she wants to provide for the iteration's termination.

In this second scenario, SM eliminates obstacles during the sprint and identifies the main impediments of the last completed sprint in a retrospective meeting (Sutherland et al., 2014). So, DT receives resources r_1 to overcome the beginning of the sprint, and later it gets additional resources r_2 to deal with the sprint's difficulties (see Figure 3.4). So, the total quantity of resources is $r = r_1 + r_2$, while the resources profiles $\overline{r} = (r_1, r_2)$ denotes a strategy of the scrum master. Hence, SM's strategy comprehends a two-stages intervention.

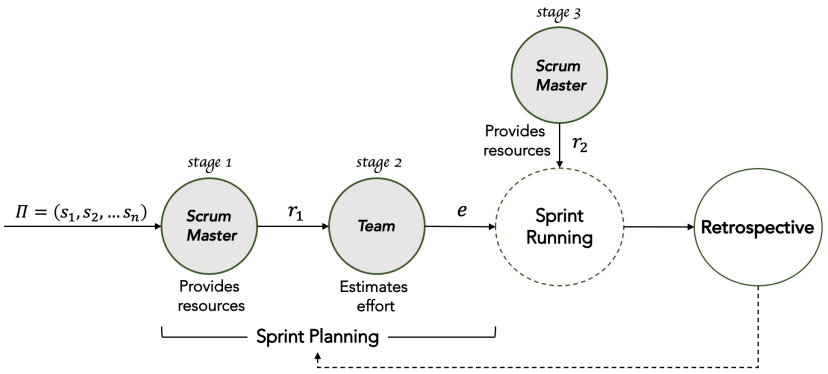

Figure 3.4: Scenario 2.

Source: Own elaboration.

Based on the sequence of events of this second scenario, the backward induction process starts by the last stage. So, we first compute the additional resources r_2 that SM provides for the project's termination in the last stage. Later, we search for the equilibrium effort that DT exerts in the stage 2. Finally, we compute the initial resources r_1 that SM provides at the iteration's beginning.

Resources at stage 3

Taking into account that $r = r_1 + r_2$, the SM's benefit function in this third scenario is

$$\pi_{SM}(r_1, r_2, e) = (K + r_1 + r_2)\ \sqrt{G + e}\ - \frac{(K + r_1 + r_2)^2}{f\ \tau}.$$

Given resources r_1 and effort e, the SM's decision problem at stage 3 is

$$\max_{r_2 \in \mathbb{R}_+} \pi_{SM}(r_1, r_2, e,\ r_2).$$

The first-order condition states that SM's optimal strategies in the final stage are the solutions of equation $\partial \pi_{SM} / \partial r_2 = 0$. The first derivative is

$$\frac{\partial \pi_{SM}}{\partial r_2} = \sqrt{G + e}\ - \frac{2\ (K + r_1 + r_2)}{f\tau}.$$

By solving the previous expression equal to zero, the critical point of π_{SM} is

$$r_2^* = \frac{f\tau\sqrt{G + e}}{2} - K - r_1. \tag{3.8}$$

Noticing that the second derivative is $\partial^2 \pi_{SM} / \partial {r_2}^2 = -2/f\tau$, we conclude that the SM's benefit function reaches a maximum at point r_2^*. In other words, r_2^* is the equilibrium strategy of SM to provide additional resources at stage 3.

Effort at stage 2

Moving backwards, we substitute r_2^* into the benefit function of DT to get the equilibrium at stage 2. Given r_1, the DT's decision problem at stage 2 is

$$\max_{e \in A_{DT}} \pi_{DT}(r_1, r_2^*, e).$$

Considering that $r = r_1 + r_2$, the DT's benefit is

$$\pi_{DT}\left(r_1, r_2^*, e\right) = I + \left(K + r_1 + \frac{f\tau\sqrt{G+e}}{2} - K - r_1\right)(G+e) - \gamma\frac{(G+e)^2}{h}.$$

Now, we compute all the critical points of function π_{DT} to maximize the benefit of DT. First, we solve the equation $\partial \pi_{DT}/\partial e = 0$ to get all critical points. It is easy to check that we get the same equation as in the subsection 3.5.1, i.e., we have that

$$\frac{\partial \pi_{DT}}{\partial e} = \frac{3f\tau\sqrt{(G+e)}}{4} - \frac{2\ \gamma\ \ (G+e)}{h}.$$

Hence, π_{DT} has a unique critical point ϵ^{*3} that is the effort strategy that we get in scenario 2, see expression (7), which is

$$\epsilon^{*3} = \frac{9h^2 f^2 \tau^2}{64\ \gamma^2} - G. \tag{3.9}$$

As in subsection 3.5.1, the second derivative of π_{DT} is

$$\frac{\partial^2 \pi_{DT}}{\partial e^2} = -\frac{2\ \gamma}{h} + \frac{3f\tau}{8\sqrt{G+e}}.$$

Then, we can follow the same reasoning as before to verify that π_{DT} reaches a local maximum at effort ϵ^{*3}, i.e., we conclude that strategy (3.9) is the equilibrium strategy for the development team at stage 2 of scenario 2.

Resources at stage 1 equilibrium

To find all the equilibria of stage 1, we substitute r_2^* and ϵ^{*3}, see expressions (3.8) and (3.9), into the scrum master benefit function (3.2). We get that

$$\begin{aligned}\pi_{SM}(r,e) &= \left(K + r_1 + \frac{f\tau\sqrt{G+e}}{2} - K - r_1\right)\sqrt{G + \frac{9h^2f^2\tau^2}{64\ \gamma^2} - G} + \dots \\ \dots &\quad - \frac{\left(K + r_1 + \frac{f\tau\sqrt{G+e}}{2} - K - r_1\right)^2}{f\ \tau} \\ &= \frac{f\tau\sqrt{G+\epsilon^{*3}}}{2}\sqrt{\frac{9h^2f^2\tau^2}{64\ \gamma^2}} - \frac{f\tau(G+\epsilon^{*3})}{4}.\end{aligned}$$

In the previous expression, we observe that SM's depends on ϵ^{*3}, which is independent of the resources provided by the scrum master. Hence, π_{SM} is independent of variables r_1 and r_2. In other words, the benefit of SM is derived from the effort that the development team exerts to cope with the sprint objectives. So, in this scenario, the scrum master behaves as a follower, which means that she embraces a **Servant Leadership** (Holtzhausen and de Klerk, 2018).

Note that SM's equilibrium strategies are resources profiles (r_1^*, r_2^*) such that $r_1^* \in \mathbb{R}_+$ and

$$r_2^* = \frac{3\ hf^2\ \tau^2}{16\ \gamma} - K - r_1^*.$$

Since there is a negative relationship between r_2^* and r_1^*, the scrum master provides fewer additional resources when she focuses on her initial intervention. However, the non-provision of resources in the first stage is also an equilibrium strategy; in this case, the additional resources r_2^* summarize the total needs of the development team. Finally, it is worth recalling that not all resources profiles (r_1^*, r_2^*) are coherent due to the previous negative relationship; that is to say, the initial resources r_1^* should imply that $r_2^* \geq 0$. The previous discussion is summarized in the following proposition.

Proposition 3.3. *Consider the interaction between SM and DT concerning the scenario 2 of the EE game. The sub-game perfect Nash equilibria are the strategies profiles*

$$\left((r_1^*, r_2^*), \epsilon^{*3}\right) = \left(\left(r_1^*, \frac{3\ hf^2\ \tau^2}{16\ \gamma} - K - r_1^*\right), \frac{9\ h^2f^2\ \tau^2}{64\ \gamma^2} - G\right),$$

for all initial resources $r_1^* \in \left[0, \frac{3\ hf^2\ \tau^2}{16\ \gamma} - K\right]$.

3.5.3 Comparison between scenarios' equilibrium

This section analyzes how players' equilibrium strategies and payoffs change as exogenous variables change. Also, we compare the agents' equilibrium behaviour in the original game and the two variations that we analyzed in subsections 3.5.1 and 3.5.2.

Proposition 3.4. ***Comparative statics in the original game.*** *Consider the equilibrium* (r^*, ϵ^*) *of the original game stated in Proposition 3.1. We have that*

(a) $\frac{\partial r^*}{\partial f}, \frac{\partial \epsilon^*}{\partial f} > 0$,

(b) $\frac{\partial r^*}{\partial \tau}, \frac{\partial \epsilon^*}{\partial \tau} > 0$,

(c) $\frac{\partial r^*}{\partial h}, \frac{\partial \epsilon^*}{\partial h} > 0$, *and*

(d) $\frac{\partial r^*}{\partial \gamma}, \frac{\partial \epsilon^*}{\partial \gamma} < 0$.

Proof. See Appendix A. □

Proposition 3.5. ***Comparative statics in the scenario 1.*** *By Proposition 3.2, the profile* (r^{*2}, ϵ^{*2}) *is the sub-game perfect Nash equilibrium of Scenario 1. Concerning such equilibrium strategies, the following relationships hold for the parameters* f, h, τ *and* γ.

(a) $\frac{\partial r^{*2}}{\partial f}, \frac{\partial \epsilon^{*2}}{\partial f} > 0$,

(b) $\frac{\partial r^{*2}}{\partial \tau}, \frac{\partial \epsilon^{*2}}{\partial \tau} > 0$,

(c) $\frac{\partial r^{*2}}{\partial h}, \frac{\partial \epsilon^{*2}}{\partial h} > 0$, *and*

(d) $\frac{\partial r^{*2}}{\partial \gamma}, \frac{\partial \epsilon^{*2}}{\partial \gamma} < 0$.

Proof. See Appendix A. □

Proposition 3.6. ***Comparative statics in the scenario 2.*** *Let* $(r_1^*, r_2^*, \epsilon^{*3})$ *be the general form of sub-game perfect Nash equilibria of Scenario 2, whenever assumptions in Proposition 3.3 hold. Concerning such equilibrium strategies, the following relationships hold for the parameters* f, h, τ *and* γ.

(a) $\frac{\partial r_1^*}{\partial f} = 0,\ \frac{\partial r_2^*}{\partial f},\ \frac{\partial \epsilon^{*3}}{\partial f} > 0,$

(b) $\frac{\partial r_1^*}{\partial \tau} = 0,\ \frac{\partial r_2^*}{\partial \tau},\ \frac{\partial \epsilon^{*3}}{\partial \tau} > 0,$

(c) $\frac{\partial r_1^*}{\partial h} = 0,\ \frac{\partial r_2^*}{\partial h},\ \frac{\partial \epsilon^{*3}}{\partial h} > 0,$ *and*

(d) $\frac{\partial r_1^*}{\partial \gamma} = 0,\ \frac{\partial r_2^*}{\partial \gamma},\ \frac{\partial \epsilon^{*3}}{\partial \gamma} < 0.$

Proof. See Appendix A. □

Propositions 3.4, 3.5 and 3.6 illustrate a positive relationship between equilibrium strategies and *DT*'s experience (h), *SM*'s flexibility (f), and the environmental stability τ. In other words, effort and resources increase as the previous parameters also increase, which is an intuitive result since it reflects the non-existence of surrounding factors that slow down the pace of work. Conversely, there is a negative relationship between the *sprint backlog* complexity (γ) and players' behaviour at equilibrium, which means that effort and resources decrease as the complexity of user stories increases. The last relationship may reflect the fact that complexity is closely related to difficulty (Ziauddin and Zia, 2012), i.e., both agents diminish their actions as complexity increases since the project's task appear impossible to finish with their current skills and resources. So, complexity raises the sprint costs, charging both the *DT* and the *SM*.

In addition, r_1^* is independent of all exogenous parameters, which means that *SM* is free to choose the quantity of initial resources. For example, the scrum master can provide all the necessary resources to support the development team before the sprint begins $\left(r_1^* = \frac{3hf^2\tau^2}{16\gamma} - K\right)$, or to wait until the development team exerts effort to provide the resources that *DT* needs ($r_1^* = 0$). Interestingly, the development team exerts the same effort in the game's three versions when $r_1^* = 0$.

Proposition 3.7. *Consider that* $r_1^* = 0$. *The equilibrium effort strategy is the same in the three versions of the EE game, i.e., we have that*

$$\epsilon^* = \epsilon^{*2} = \epsilon^{*3}.$$

Proof. See Appendix A. □

Now, we analyze which scenario is better for each player. Although equilibrium effort remains the same in the three versions of the game, when certain conditions are fulfilled, the DT's benefit changes since resources are not the same in the three game's versions. Particularly, the DT gets its largest benefit at equilibrium in the original game.

Theorem 3.1. *If parameters f, h, τ and γ are positive, the Development Team gets its higher benefit in the original EE game, in comparison with scenarios 1 and 2.*

Proof. Firstly, we compare the benefits that DT gets in the basic EE game with those that he gets in the first scenario. From proposition 3.1, we know the players' strategies in the original EE game are r^* and ϵ^*. By evaluating them in the DT's payoff function, we get that

$$\begin{aligned} \pi_{DT}^{O} &= \pi_{DT}(r^*, \epsilon^*) \\ &= I + \left(\frac{9\ \tau^2\ h\ f^2}{32\ \gamma}\right)\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right) - \gamma\frac{\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right)^2}{h} \\ &= I + \frac{81\ \tau^4\ h^3\ f^4}{2048\ \gamma^3} - \frac{81\ \tau^4\ h^3\ f^4}{4096\ \gamma^3} \\ &= I + \frac{81\ \tau^4\ h^3\ f^4}{4096\ \gamma^3}. \end{aligned} \tag{3.10}$$

Now, from Proposition 3.2, the equilibrium strategies in scenario 1 are r^{*2} and ϵ^{*2}. Then, we replace them in the DT's benefit function.

$$\begin{aligned} \pi_{DT}^{1} &= \pi_{DT}(r^{*2}, \epsilon^{*2}) \\ &= I + \left(\frac{3\ \tau^2\ h\ f^2}{16\ \gamma}\right)\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right) - \gamma\frac{\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right)^2}{h} \\ &= I + \frac{27\ \tau^4\ h^3\ f^4}{1024\ \gamma^3} - \frac{81\ \tau^4\ h^3\ f^4}{4096\ \gamma^3} \\ &= I + \frac{27\ \tau^4\ h^3\ f^4}{4096\ \gamma^3}. \end{aligned} \tag{3.11}$$

Note that $81/4096 > 27/4096$. Thus, we conclude that

$$\pi_{DT}^{O} > \pi_{DT}^{1}.$$

Now, we compare DT's benefits in the first scenario with those that it gets in the second scenario. Concerning this game's variation, Proposition 3.3 shows the equilibrium profile $(r_1^*, r_2^*, \epsilon^{*3})$. Next,

we substitute equilibrium strategies into the DT's benefit function to compare the development team benefit at different games. Note that

$$\begin{aligned}
\pi^2_{DT} &= \pi_{DT}(r^*_1, r^*_2, \epsilon^{*3}) \\
&= I + (K + r^*_1 + r^*_2)(G + e) - \gamma \frac{(G+e)^2}{h} \\
&= I + \left(K + r^*_1 + \frac{3\ \tau^2\ h\ f^2}{16\ \gamma} - K - r^*_1\right)\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right) - \gamma \frac{\left(\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}\right)^2}{h} \\
&= I + \left(\frac{27\ \tau^4\ h^3\ f^4}{1024\ \gamma^3}\right) - \frac{81\ \tau^4\ h^3\ f^4}{4096\ \gamma^3} \\
&= I + \frac{27\ \tau^4\ h^3\ f^4}{4096\ \gamma^3}.
\end{aligned} \tag{3.12}$$

Hence, if $\pi^2_{DT} > \pi^O_{DT}$, we have that

$$I + \frac{27\ \tau^4\ h^3\ f^4}{4096\ \gamma^3} \quad > \quad I + \frac{81\ \tau^4\ h^3\ f^4}{4096\ \gamma^3}$$

Note that the previous expression is not true since $27/4096 < 81/4096$; i.e., we conclude that

$$\pi^2_{DT} < \pi^O_{DT}.$$

□

Below, we compare the scrum master benefits in the three interactions that we analyze. The following theorem establishes that scenarios 1 and 2 offer higher payoffs to SM with respect to the original EE game.

Theorem 3.2. *If parameters f, h, τ and γ are positive, the scrum master gets a higher payoff and provide less additional resources in scenarios 1 and 2 than in the original EE game. Her benefits in scenario 1 exceeds those received in scenario 2, when $r^*_1 > 0$.*

Proof. Firstly, we compare SM's benefits in the original EE game with those he gets in scenario 1. From proposition 3.1, r^* and ϵ^* are the equilibrium startegies at the original game. By evaluating

them in the SM's payoff function, we have that

$$\begin{aligned} \pi_{SM}^{O} &= \pi_{SM}(r^*, \epsilon^*) \\ &= \left(\frac{9\ \tau^2\ h\ f^2}{32\ \gamma}\right)\sqrt{\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}} - \frac{\left(\frac{9\ \tau^2\ h\ f^2}{32\ \gamma}\right)^2}{f\ \tau} \\ &= \frac{27\ \tau^3\ h^2\ f^3}{256\ \gamma^2} - \frac{81\ \tau^3\ h^2\ f^3}{1024\ \gamma^2} \\ &= \frac{27\ \tau^3\ h^2\ f^3}{1024\ \gamma^2}. \end{aligned} \tag{3.13}$$

Concerning scenario 1, the equilibrium strategies are r^{*2} and ϵ^{*2}; see Proposition 3.2. We substitute the previous strategies into the scrum master benefit function; we get that

$$\begin{aligned} \pi_{SM}^{1} &= \pi_{SM}(r^{*2}, \epsilon^{*2}) \\ &= \left(\frac{3\ \tau^2\ h\ f^2}{16\ \gamma}\right)\sqrt{\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}} - \frac{\left(\frac{3\ \tau^2\ h\ f^2}{16\ \gamma}\right)^2}{f\ \tau} \\ &= \frac{9\ \tau^3\ h^2\ f^3}{128\ \gamma^2} - \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2} \\ &= \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2}. \end{aligned} \tag{3.14}$$

Since $9/256 > 27/1024$, we conclude that

$$\pi_{SM}^{1} > \pi_{SM}^{O}.$$

In other words, the scrum master gets a larger benefit when she behaves as a follower.

Now, we proceed to compare the scrum master benefits in scenarios 1 and 2. In scenario 2, the profile of equilibria strategies are represented as $(r_1^*, r_2^*, \epsilon^{*3})$, see Proposition 3.3. By substituting such strategies into the benefit function, we get that

$$\begin{aligned} \pi_{SM}^{2} &= \pi_{SM}(r_1^*, r_2^*, \epsilon^{*3}) \\ &= \left(r_1^* + \frac{3\ \tau^2\ h\ f^2}{16\ \gamma}\right)\sqrt{\frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2}} - \frac{\left(r_1^* + \frac{3\ \tau^2\ h\ f^2}{16\ \gamma}\right)^2}{f\ \tau} \\ &= \frac{3\ \tau\ h\ f}{8\ \gamma} r_1^* + \frac{9\ \tau^3\ h^2\ f^3}{128\ \gamma^2} - \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2} - \frac{(r_1^*)^2}{f\ \tau} - \frac{3\ \tau\ h\ f}{8\ \gamma} r_1^* \\ &= \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2} - \frac{(r_1^*)^2}{f\ \tau}. \end{aligned} \tag{3.15}$$

To guarantee that $\pi^2_{SM} > \pi^1_{SM}$, we have that

$$\begin{aligned} \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2} - \frac{(r_1^*)^2}{f\ \tau} &> \frac{9\ \tau^3\ h^2\ f^3}{256\ \gamma^2} \\ -\frac{(r_1^*)^2}{f\ \tau} &> 0 \\ r_1^* &< 0. \end{aligned} \tag{3.16}$$

Therefore, $\pi^1_{SM} > \pi^2_{SM}$ for all $r_1^* > 0$. □

Theorem 3.2 says that the scrum master gets a larger benefit when it provides resources at the sprint beginning. However, the previous result also establishes a condition under the SM prefers to change her role from leader to follower. It is worth recalling that the previous condition depends on parameters f, h, τ, and γ. Hence, the scrum master may choose to change the stage where she intervenes depending on such a condition. However, it is worth noticing that equilibrium effort does not change if the scrum master provides resources at the initial or last stages.

3.6 Numerical examples

In this section we present some numerical examples to illustrate the changes of equilibrium strategies (r^*, e^*) and benefits (π_{SM} and π_{DT}) when exogenous variables change. Specifically, we consider positive values of variables f, h, τ and γ since they summarize internal and external conditions related with the SM and DT interaction (see Theorems 3.1 and 3.2). To simplify the analysis, we first consider that $I = G = K = 0$, while Table 3.2 shows the values for other parameters.

3.6.1 Changes in DT's strategy and payoff

By assuming that conditions of Propositions 3.4, 3.5 and 3.6 hold, we have that DT's strategy increases as h, f and τ increase. Also, DT's effort decreases as γ increases. As we can observe in Table 3.3 and Figure 3.5, the previous behaviour hold in the three game's versions. We show that DT's strategy and payoff rise when agents' have better skills, specifically, when their types f and h go from 0.7 to 1). Analogously, when the environment is more stable (τ goes from 0.85 to 1), effort and resources increase at equilibrium. Conversely, when we have a more complex sprint backlog, with γ between

Table 3.2: Values assigned to exogenous variables

Parameters	Values	Interpretation
h	0.7, 1	Above average, High experience
f	0.7, 1	Above average, High flexibility
γ	0.1, 0.5	Low and Medium sprint backlog complexity
τ	0.85, 1	Medium and High environmental stability

Source: Own elaboration.

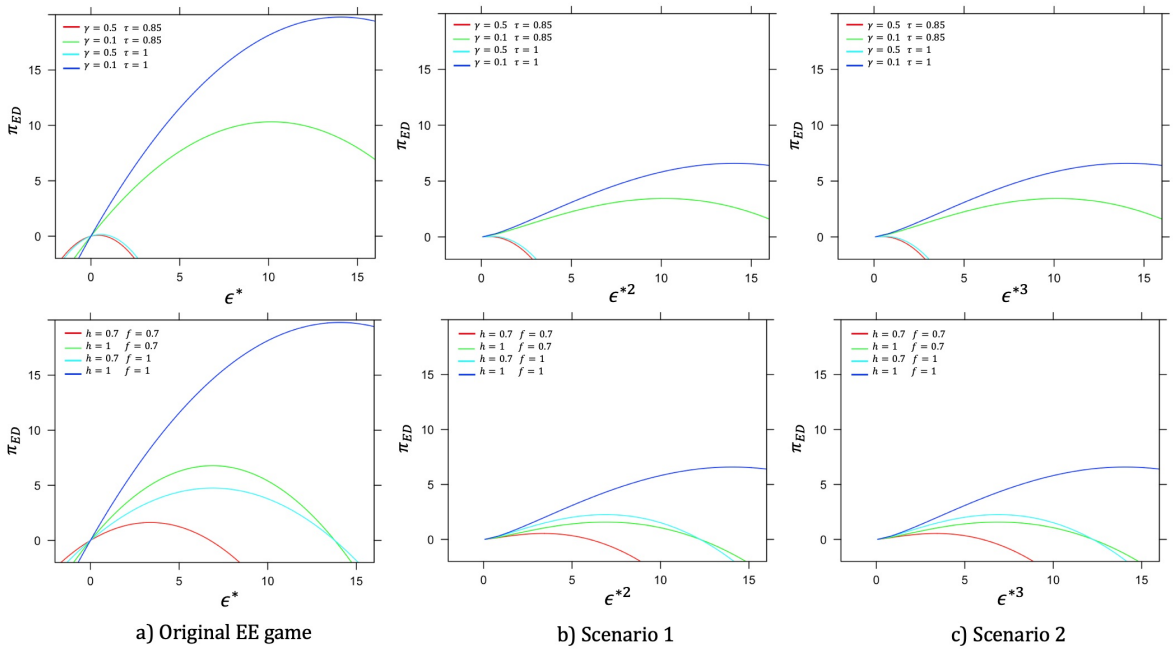

Figure 3.5: Changes in ϵ^* and π_{DT} as h, f, γ and τ change.

Source: Own elaboration.

Table 3.3: DT's strategy and payoffs as h, f, γ and τ change

		Original EE game		Scenario 1		Scenario 2	
h	f	ϵ^*	π^O_{DT}	ϵ^{*2}	π^1_{DT}	ϵ^{*3}	π^2_{DT}
0.7	0.7	3.3764	1.6285	3.3764	0.5428	3.3764	0.5428
1	0.7	6.8906	4.7480	6.8906	1.5826	6.8906	1.5826
0.7	1	6.8906	6.7829	6.8906	2.2609	6.8906	2.2609
1	1	14.0625	19.7753	14.0625	6.5917	14.0625	6.5917
γ	τ	ϵ^*	π^O_{DT}	ϵ^{*2}	π^1_{DT}	ϵ^{*3}	π^2_{DT}
0.5	0.85	0.4064	0.0825	0.4064	0.0275	0.4064	0.0275
0.5	1	0.5625	0.1582	0.5625	0.05273	0.5625	0.0527
0.1	0.85	10.1601	10.3228	10.1601	3.4409	10.1601	3.4409
0.1	1	14.0625	19.7753	14.0625	6.5917	14.0625	6.5917

Source: Own elaboration.

0.1 and 0.5, DT's strategy and payoff drop significantly. Nevertheless, as Theorem 3.1 states, we can observe that DT's payoffs are higher in the original EE game than those benefits unfolded in scenarios 1 and 2. This may change in scenario 2, when SM chooses to provide enough resources at the beginning of the sprint, i.e. $r_1^* = \frac{3hf^2\tau^2}{16\gamma} - K$.

3.6.2 Changes in SM's strategy and payoff

Concerning the scrum master, Propositions 3.4, 3.5 and 3.6 establishes the conditions under the quantity of resources provided by SM increases as parameters h, f and τ also increase. Also, such propositions establish that resources decrease as γ increases. In Table 3.4 we present numerical examples that satisfy the conditions mentioned above, while Figure 3.6 illustrates the equilibrium strategies in the three game's versions. Note that SM's strategy preserves this behaviour even if $r_1^* = 0$. However, as it happens with the DT's equilibrium behaviour, the SM's strategy and payoff increase when the agent's types h and f go from 0.7 to 1, that represents high flexibility and advanced

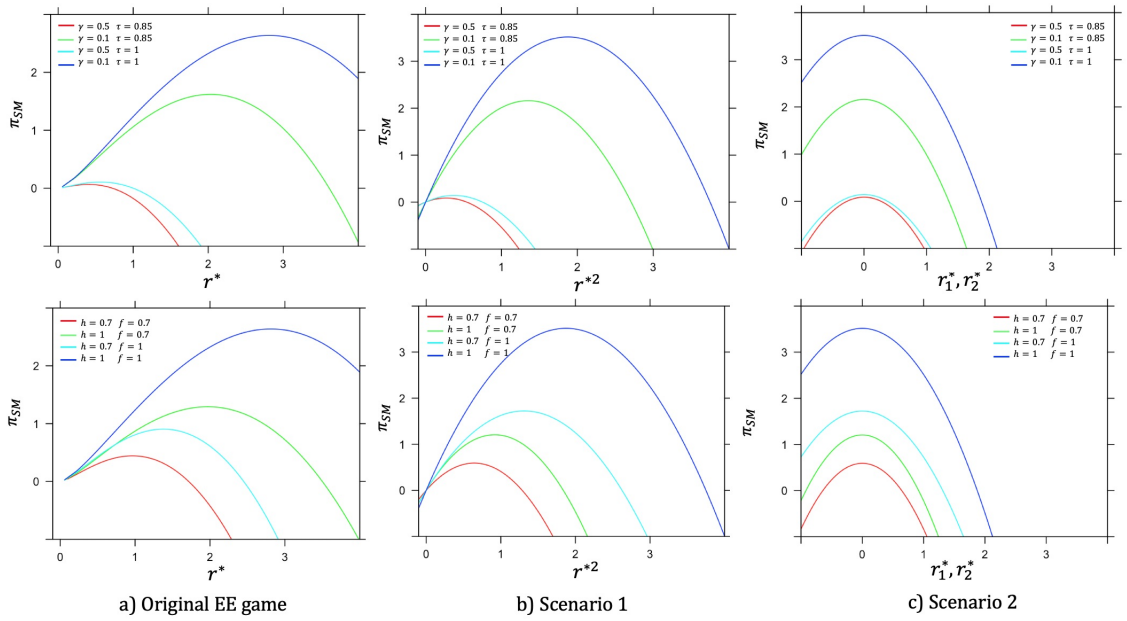

Figure 3.6: Changes in r^* and π_{SM} as h, f, γ and τ change.

Source: Own elaboration.

skills, and in a stability with $\tau \in [0.85, 1]$. Inversely, as the sprint backlog complexity γ moves from 0.1 to 0.5, the scrum master provides fewer resources while her benefit reduces.

Nevertheless, as Theorem 3.2 states, we can observe that SM's payoffs are higher in scenarios 1 and 2 than those produced in the original EE game. Meanwhile, the SM's equilibrium strategy provides more additional resources in the EE original game than in scenarios 1 and 2. However, the previous statement is not always true in scenario 2 since the SM may provide enough resources at the beginning of the sprint; as Theorem 3.1 points, the initial resources r_1^* discourages a second intervention of the SM. Therefore, SM can choose from multiple strategies to scale the sprint results in scenario 2.

3.7 Conclusions

This research proposes a game theoretical framework to analyze effort estimation in a Scrum context where players have complete information about each other. Here, the SM and the DT know the other's type, meaning that attributes and capabilities are common knowledge. So, we focus on analyzing

Table 3.4: SM's strategy and payoffs as h, f, γ and τ change

		Original EE game		Scenario 1		Scenario 2	
h	f	r^*	π^O_{SM}	r^{*2}	π^1_{SM}	r^*_1, r^*_2	π^2_{SM}
0.7	0.7	0.9646	0.4431	0.6431	0.5908	0, 0.6431	0.5908
1	0.7	1.3781	0.9043	0.9187	1.2058	0, 0.9187	1.2058
0.7	1	1.9687	1.2919	1.3125	1.7226	0, 1.3125	1.7226
1	1	2.8125	2.6367	1.875	3.5156	0, 1.875	3.5156
γ	τ	r^*	π^O_{SM}	r^{*2}	π^1_{SM}	r^*_1, r^*_2	π^2_{SM}
0.5	0.85	0.4064	0.0647	0.2709	0.0863	0, 0.2709	0.0863
0.5	1	0.5625	0.1054	0.375	0.1406	0, 0.375	0.1406
0.1	0.85	2.0320	1.6192	1.3546	2.1590	0, 1.3546	2.1590
0.1	1	2.8125	2.6367	1.875	3.5156	0, 1.875	3.5156

Source: Own elaboration.

effort exertion in the strategic interaction of a mature team, having major importance for companies where consolidated insourcing practices are part of their competitive strategy. In our game-theoretical model, the types of SM and DT represents flexibility and experience. Also, we consider the impact of exogenous variables such as the sprint backlog complexity, environmental stability, basic infrastructure, and DT fixed expense. As the scrum master is in charge of providing resources to the team, we investigate the magnitude of resources provided and the effort level at equilibrium. Later, we look at the relationship between the equilibrium's benefits and the exogenous variables.

Additionally, our modelling approach allows us to analyze players' behaviour by considering two game's variations. Notice that the SM plays first as a leader, and then DT is the follower in the original EE game. In the first variation, scenario 1, we consider that DT plays first now, transforming SM into the follower. In scenario 2, our second interaction variation, SM plays first, then DT continues, and SM plays again.

In our main results, Theorems 3.1 and 3.2, we state that the development team gets its highest payoff

in the original EE game, while the scrum master gets her highest payoff in scenarios 1 and 2. Also, the resources provided in the first and second scenarios are fewer than those provided in the original EE game. Consequently, in scenarios 1 and 2, SM gets a higher payoff since she supplies fewer resources. In other words, the scrum master has incentives to change their intervention role during a project's development; in this case, the follower role provides the largest benefits. On the contrary, although equilibrium effort does not change in the three games, the development team prefers the original EE game.

Also, our main results point out that agents get the largest benefit when they behave as followers, which is in an interesting result since the opposite holds in sequential games like the classical Stackelberg model. Moreover, players' benefit diminishes whenever the sprint backlog complexity increases or lacks environmental stability. We also observe that the sprint backlog complexity is the exogenous variable that most impacts the sprint's results. Numerical examples show that DT's payoff drops dramatically as sprint backlog complexity increases, regardless of the scenario.

Relating to scenario 2, as SM plays twice, first and third place, SM starts setting no additional resources, and later she provides the necessary additional resources. So, the SM's strategy consists of two parts. Notwithstanding, the SM's strategy differs between scenarios. Meanwhile, DT's strategies and payoffs are the same in all three scenarios. Then, this second scenario does not raise effort or resources; nevertheless, it maintains players' benefits. These can be seen in numerical examples.

In conclusion, we can say that the sprint results are favoured when the players interact after estimating the effort, as happens in scenarios 1 and 2. This allows optimizing the amount of additional resources necessary to achieve the sprint objectives, according to the previously estimated effort. This is in line with the scrum master description as a Servant Leader, stating that she encourages followers' autonomy and makes room for decision-making (Holtzhausen and de Klerk, 2018). Then, a scrum master should become the follower for being more a facilitator leader, while the team commits as a leader in the software development. We think this could provide useful insights for agile software development environments.

It is worth emphasizing that our approach deals with the sprint backlog complexity and environmental stability during software development inside an iteration. So, in futures works, we pretend to extend

the game-theoretical model to dynamic frameworks where it is important to differentiate between user stories as well as updating the iteration velocity. Also, in future works, we pretend to focus on analyzing the incomplete information case, i.e., where players are unaware of each other's attributes and abilities.

Chapter 4

EE in agile global software development with offshoring teams

4.1 Introduction

Effort estimation (EE) is critical to plan software projects where accurate estimates are required for successful product delivery (Mohagheghi and Jørgensen, 2017; Jørgensen, 2016a). This matters more than ever, since a growing demand for software is expected by the different social and economic sectors over the world (Brem et al., 2021). Some of these applications need experienced and skilled developers with knowledge in diverse technologies. Hence, Global Software Development (GSD) is attractive for companies to access distant talents with diverse backgrounds (Wickramaarachchi and Lai, 2017). Nevertheless, EE is difficult due to people and processes issues (Popli and Chauhan, 2014). Moreover, while agile methodologies promote close collaboration between co-located teams; global development seeks to incorporate human resources from different organizational and geographical locations. So, EE in merged environments such as Agile Global Software Development (AGSD) represents a greater challenge. In addition, studies on EE within AGSD are scarce.

In fact, teamwork becomes paramount for EE in AGSD. As happens in agile development, EE lies

in the relationship of mutual dependence between the project leader and the development team to accomplish the project's objectives. However, they may have different objectives (Wachnik, 2015; Moe et al., 2012). In Scrum, the most widely used agile methodology (Digital.ai., 2020; Mutiullah et al., 2018), the scrum master (SM) focuses on producing the highest busines value, while the development team (DT) wants to maximize its profits by exerting a minimum effort. Furthermore, in global development environments, DT consists of several teams with base in different locations. Hence, AGSD accounts for a communication overload, as well as coordination and collaboration concerns (Qureshi and Alsulami, 2014) that compromise EE precision, if they are not handled properly. Britto et al. (2015) found that almost 55% of time the effort is underestimated in AGSD contexts; also, 27.5% of their study's answers reported that EE accuracy is affected when team members do not know each other. It is worth mentioning that SM have limited information regarding the outsourced-offshore teams' skills, since SM has not worked with them before. Physical distance, as well as cultural and language differences, also make difficult to access information about offshore teams, even when they are insourced (Alzoubi and Gill, 2014). In the same way, offshore teams have incomplete information on SM's skills. Thus, agency problems may emerge such as moral hazard (Eisenhardt, 1989), leading to an unrealistic EE. In this regard, some studies have found that developers engage in behaviours such as *decision hijacking*, *free-raider* and *gold plating* while sprint planning meetings are running (Moe et al., 2010; Gulliksen Stray et al., 2011; Moe, 2013). Moe et al. (2010) explain that team members follow their own plan without discuss it with their partners, and focus on perfecting what they consider primordial by their alone. In addition, outsourced developers could be working in several projects at the same time with different incentives by each project (Conoscenti et al., 2019); this may impact developers' decisions. So, team members might give biased effort estimations by their personal or professional priorities. Keil (2005) found that principal-agent issues become evident in business software with outsourcing and offshoring schemes, where divergent interests lead to opportunistic behaviours, such as hidden characteristics, intentions and actions.

In this research, we study the impact of leader-teams interaction during agile software development in a global environment. We based on the principal-agent problem (Eisenhardt, 1989); we consider that the SM (principal) guides the DT (agent) to develop a software project in the presence of imperfect

information. Considering that SM is in charge of providing resources to DT, we investigate the magnitude of resources expended depending on the SM type and the impact of the DT type on effort estimations. Since the parties have limited information about their knowledge and skills, we model this interaction using a sequential game-theoretical approach of incomplete information. As far as our knowledge, we are the first ones that analyze EE in an AGSD context through the lens of the Game Theory. In this sense, we consider that this study is a novel contribution to the EE in AGSD literature.

This chapter is organized into six sections. In the second section, we show basic concepts on GDS and how EE is carried out in AGSD. Next, in the third section, we present the game-theoretical model that establishes effort in a AGSD context. Then, we compute the Bayesian sub-game perfect Nash equilibrium in the fourth section, and we show some numerical examples in the fifth section. Finally, we discuss the conclusions and future works.

4.2 Effort estimation in agile global software development

Companies configure diverse schemes to build software at lower cost and higher quality; so, GSD is an attractive framework since brings benefits such as reduction in costs, project cycle time and availability of a large and skilled labor pool (Yadav, 2016). Here, team members sourcing is determined by physical and legal dimensions (Šmite et al., 2014). Thus, offshore-insourcing means that a company moves the software development to a branch, while offshore-outsourcing is when the software development is moved to an external third party (Britto, 2015). So, developing under offshore schemes involves distributing the work to different locations. Of course, distributed global development faces coordination and communication challenges that agile methodologies intend to overcome them (Qureshi and Alsulami, 2014; Hossain et al., 2009; Cristal et al., 2008; Sutherland et al., 2007). Hence, the increasingly interest in the AGSD. Nevertheless, EE process in AGSD has not been studied enough and claims for a systematic method to resolve dependences between the parties involved (Wickramaarachchi and Lai, 2017). Britto (2015) proposed that effort can be estimated in a centralized or a distributed way, for AGSD environments. In a centralized EE, the main site estimates all the global effort. Meanwhile, in a distributed EE, each site estimates its own effort and then, one of the sites combines estimations.

We focus on analyzing EE in a distributed setting, where an offshore team establishes its EE to send it to the onsite team, where the SM is located. See figure 4.1.

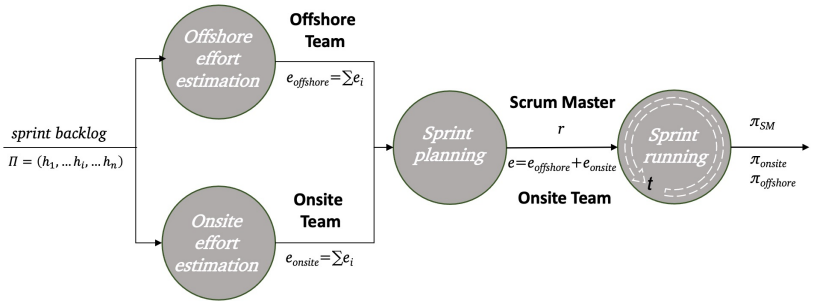

Figure 4.1: The sequence of events around a Distributed EE setting.

Source: Own elaboration.

Given the sprint backlog, offshore team and onsite team produce their own EE, taking care that their activities are not dependent between them. Note that, in a Distributed Effort Estimation (DEE) setting, both the DT and the SM must pay communication, coordination and collaboration costs, that are additional to software development effort.

4.3 The model

In this research, we focus on analyzing a generic Scrum iteration in GSD. So, we consider the interaction between the scrum master who is at onsite and one offshore development team during the development of a single sprint.

The set of agents in the offshore EE game is $J = \{SM,\ DT\}$ where SM is the scrum master and DT is the offshore development team. Given that scrum teams self-organize, DT members share goals and make decisions collectively (Srivastava and Jain, 2017); thus, we assume DT makes decisions as a single player.

Once the *sprint backlog* is planned under a GSD setting, its successful completion also depends on how agents in J behave. We consider that each agent is characterized by a certain type of behaviour. In the case of the scrum master, we assume that SM behaves as a Servant leader (Holtzhausen and

de Klerk, 2018) that is always willing to provide support to DT by removing obstacles and conflicts (Villegas Gómez et al., 2016; Srivastava and Jain, 2017). Therefore, there is a unique SM type.

Regarding the offshore team, DT calculates how much support he will need to build the *sprint backlog* based on his knowledge and previous experience in GSD (Hidayati et al., 2020). Then, the DT's type h is the vector $h = (h_r, h_c)$, where h_r represents the resources for software development (SD) that DT calculates to receive, while h_c is the communication expense that DT considers necessary. The set of all possible types of DT is $H \subseteq \mathbb{R}^2$. We assume this information is private, i.e., SM does not know the DT's type, which is the realization of the random vector $H = (H_r, H_c)$. We consider that H follows a common knowledge probability function $z : h_r \times h_c \to [0, 1]$.

4.3.1 The game

In the offshore EE game exists incomplete information due to SM does not know DT's type. This circumstance encourages the players behave strategically to get the highest possible payoff. The interaction between SM and DT to estimate the offshore effort of a given *sprint backlog* is a three-stage game. See figure 4.2.

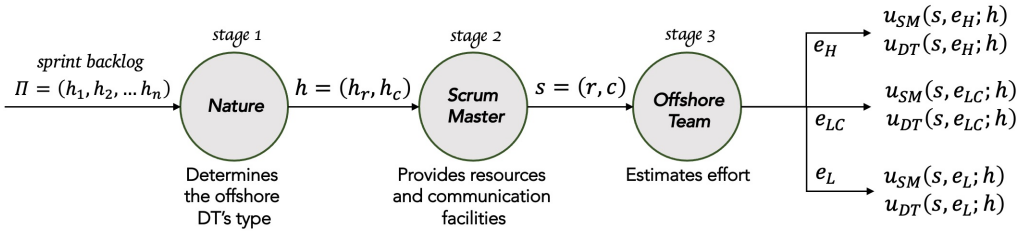

Figure 4.2: Offshoring EE game.

Source: Own elaboration.

At the first stage, nature determines the offshore DT's type h through a probability distribution z, which is observed only by DT, but not by SM. Nevertheless, SM knows that DT could take on high or low effort, the latter either in communication or overall in SD.

At the second stage, SM sets the supports' vector $s = (r,\ c)$ where r are the SD resources and c are the communication facilities that she wants to provide for the iteration's development. Thus, the

set of all SM's actions is $A_{SM} = \{(r,\ c) \in \mathbb{R}^2 | r,\ c \geq 0\}$. Both players, SM and DT observe the supports' vector s.

Finally, at the third stage, the offshore DT estimates the effort. Then, the offshore DT establishes the effort e to develop the *sprint backlog*. Here, DT has three effort alternatives e; they are high effort e_H, low communication effort e_{LC} and overall low effort e_L. Hence, the set of all possible DT's actions is $A_{DT} = \{e_H, e_{LC}, e_L\}$. After the sprint ends, agents' payoffs take place as $u_{SM}(s, e; h)$ and $u_{DT}(s, e; h)$.

The preferences of the players map an action vector $(s,\ e)$ into a real number, i.e. $u_{SM} : A_{SM} \times A_{DT} \rightarrow \mathbb{R}$ and $u_{DT} : A_{SM} \times A_{DT} \rightarrow \mathbb{R}$. We think that SM's benefits are produced by the provided resources and the DT's effort. Likewise, we consider that SM faces the same costs marked out in assumption 3.1, but excluding the DT's type which is unknown to SM in GSD. However, SM's costs increase as DT demands more supports to exert a higher effort. So, the SM's payoff function is the following

$$u_{SM}(s, e; h) = \begin{cases} (rc)\, e - \frac{(rc)^2}{\tau} & si\ e = e_H, \\ (rc)\, e - \frac{(r)^2}{\tau} & si\ e = e_{LC}, \\ (rc)\, e & si\ e = e_L. \end{cases} \tag{4.1}$$

where τ measures the stability in an agile SD environment (Ziauddin and Zia, 2012).

Concerning DT, when he exerts high effort, we consider that DT treats the same benefits and costs described in assumptions 3.3 and 3.4. However, when DT exerts some level of low effort, he is not motivated by communication facilities or even, he may also not have incentives in relation to SD resources to increase his benefits. So, the DT's payoff function is the following

$$u_{DT}(s, e; h) = \begin{cases} I + (rc)\, e - \gamma e^2 & si\ e = e_H, \\ I + (r)\, e - \gamma e^2 & si\ e = e_{LC}, \\ I + e - \gamma e^2 & si\ e = e_L. \end{cases} \tag{4.2}$$

where I is the monetary income of DT and γ measures the *sprint backlog* complexity (Ziauddin and Zia, 2012).

4.3.2 The solution concept

The solution concept is based on players' strategies. Considering that SM's type is unique, the set of actions A_{SM} is also the set of SM's pure strategies, i.e. $S_{SM} = A_{SM}$. Regarding DT's pure strategies, they are decision rules that map (h, s) into actions in A_{DT}. A DT's decision rule is a function $\phi_{DT} : H \times \mathbb{R} \to A_{DT}$. The set of all decision rules of DT is S_{DT}. A profile of pure strategies is a vector (s, ϕ) and the set of all strategies' profiles of the offshore EE game is $S = S_{SM} \times S_{DT}$.

Now, remember that DT observes the actions taken by SM at the end of the second stage; here, SM plays first as a leader although without knowing the DT's type. Thus, SM has an expectation about their benefits based on the historical effort probabilities; which is common knowledge. From the above, the solution concept for the previous game is a Bayesian sub-game perfect Nash equilibrium. Firstly, DT assesses SM's actions and chooses the action that maximizes his payoff at the end of the third stage. Then, SM seek to maximize her expected utility by setting the required support at the second stage. We denote the following:

Definition 4.1. *A strategies profile $(s^*, \phi^*) \in S$ is a Bayesian sub-game perfect Nash equilibrium, if and only if*

1. $E[u_{SM}(s^*, \phi^*)] \geq E[u_{SM}(s, \phi^*)]$ *for all* $s \in A_{SM}$, *where* $E[u_{SM}]$ *is the expected utility of SM for all* $s \in S_{SM}$.
2. $u_{DT}(s^*, \phi^*; h) \geq u_{DT}(s^*, \phi; h)$ *for all* $\phi \in S_{DT}$ *and* $h \in H$.

4.4 The Bayesian Equilibria

We compute the Bayesian sub-game perfect Nash equilibria by applying a backward induction process. We begin by finding the DT's optimal decision rule at the third stage and then, the Nash equilibria support of SM at the second stage.

4.4.1 The DT's optimal decision rule

Once SM sets the support vector $s = (r,\ c)$ in the second stage, DT observes s and chooses an action from A_{DT}. For this, DT has his own requirements vector determined from his type $h = (h_r,\ h_c)$. So,

DT's decision obeys a set of rules that allow him to maximizes his payoff when the game ends. We have the following proposition about the Nash equilibria set for *DT* at the third stage.

Proposition 4.1. *There is a unique Nash equilibrium for DT at the third stage according to the decision rule*

$$\phi^*_{DT}(s,h) = \begin{cases} e_H & si\ r \geq h_r\ and\ c \geq h_c, \\ e_{LC} & si\ r \geq h_r\ and\ c < h_c, \\ e_L & si\ r < h_r. \end{cases} \tag{4.3}$$

Proof. Let s be the support vector of SM and h be the requirements vector of DT. For each decision rule $\phi_{DT}(s,h)$, we prove that $u_{DT}(\phi^*_{DT}(s,h)) \geq u_{DT}(\phi_{DT}(s,h))$.

Before calculating *DT*'s payoff for each decision rule, we compute the first-order condition and get the critical points for each effort alternative in u_{DT}.

- *Effort alternative* e_H.

  From the first order condition $\frac{\partial}{\partial_e} I + (rc)\, e - \gamma e^2$, we have one critical point $\frac{rc}{2\gamma}$.

- *Effort alternative* e_{LC}.

  From the first order condition $\frac{\partial}{\partial_e} I + (r)\, e - \gamma e^2$, we have one critical point $\frac{r}{2\gamma}$.

- *Effort alternative* e_L.

  From the first order condition $\frac{\partial}{\partial_e} I + e - \gamma e^2$, we have one critical point $\frac{1}{2\gamma}$.

The second-order condition for each effort alternative is $\frac{\partial^2}{\partial_{e^2}} = -2\gamma$, which is always negative since γ is a positive parameter. Therefore, the above critical points maximize the *DT*'s payoff for any effort alternative.

Now, for the decision rule determined by $r \geq h_r$ *and* $c \geq h_c$, we have that the *DT*'s payoff is

- with high effort is $u_{DT}(s, e_H; h) = I + (rc)\frac{rc}{2\gamma} - \gamma\left(\frac{rc}{2\gamma}\right)^2 = I + \frac{r^2c^2}{4\gamma}$,
- with low communication effort is $u_{DT}(s, e_{LC}; h) = I + (rc)\frac{r}{2\gamma} - \gamma\left(\frac{r}{2\gamma}\right)^2 = I + \frac{r^2(2c-1)}{4\gamma}$,
- with overall low effort is $u_{DT}(s, e_L; h) = I + (rc)\frac{1}{2\gamma} - \gamma\left(\frac{1}{2\gamma}\right)^2 = I + \frac{2rc-1}{4\gamma}$.

We conclude that $u_{DT}(\phi^*_{DT}) = u_{DT}(s, e_H; h) \geq u_{DT}(s, e_{LC}; h)$ and $u_{DT}(s, e_H; h) \geq u_{DT}(s, e_L; h)$.

Following with the decision rule determined by $r \geq h_r$ *and* $c < h_c$, we have that the DT's payoff is

- with high effort is $u_{DT}(s, e_H; h) = I + (r)\frac{rc}{2\gamma} - \gamma\left(\frac{rc}{2\gamma}\right)^2 = I + \frac{r^2(2c-c^2)}{4\gamma}$,
- with low communication effort is $u_{DT}(s, e_{LC}; h) = I + (r)\frac{r}{2\gamma} - \gamma\left(\frac{r}{2\gamma}\right)^2 = I + \frac{r^2}{4\gamma}$,
- with overall low effort is $u_{DT}(s, e_L; h) = I + (r)\frac{1}{2\gamma} - \gamma\left(\frac{1}{2\gamma}\right)^2 = I + \frac{2r-1}{4\gamma}$.

We conclude that $u_{DT}(\phi^*_{DT}) = u_{DT}(s, e_{LC}; h) \geq u_{DT}(s, e_H; h)$ and $u_{DT}(s, e_{LC}; h) \geq u_{DT}(s, e_L; h)$.

Concerning the decision rule determined by $r < h_r$, the DT's payoff is

- with high effort is $u_{DT}(s, e_H; h) = I + \frac{rc}{2\gamma} - \gamma\left(\frac{rc}{2\gamma}\right)^2 = I + \frac{2rc-r^2c^2}{4\gamma}$,
- with low communication effort is $u_{DT}(s, e_{LC}; h) = I + \frac{r}{2\gamma} - \gamma\left(\frac{r}{2\gamma}\right)^2 = I + \frac{2r-r^2}{4\gamma}$,
- with overall low effort is $u_{DT}(s, e_L; h) = I + \frac{1}{2\gamma} - \gamma\left(\frac{1}{2\gamma}\right)^2 = I + \frac{1}{4\gamma}$.

We conclude that $u_{DT}(\phi^*_{DT}) = u_{DT}(s, e_L; h) \geq u_{DT}(s, e_H; h)$ and $u_{DT}(s, e_L; h) \geq u_{DT}(s, e_{LC}; h)$.

□

The decision rule ϕ^*_{DT} allows maximize the DT's payoff by applying only one effort alternative; that is expression 4.3 is the optimal strategy for DT. Thus, when $r \geq h_r$ *and* $c \geq h_c$, DT has no reason to exert low effort; meanwhile, when $r \geq h_r$ *and* $c < h_c$, DT has no incentives to strive in communication. In contrast, when $r < h_r$, DT has no motives to increase his effort at all. Therefore, ϕ^*_{DT} is an equilibrium decision rule for DT.

4.4.2 The support at equilibrium

Following the backward induction process, we compute the Nash equilibria support of SM at the second stage. Given the decision rule ϕ^*_{DT}, we know the Nash equilibria of DT at the third stage, which depends on the requirements vector $h = (h_r,\ h_c)$ of DT. However, SM does not know $h = (h_r,\ h_c)$; hence, SM is not sure about her benefits. However, SM has an expectation about their benefits based on the historical effort probabilities; which is common knowledge. Historical data bases of effort estimation follow a probability distribution that is uniformly distributed on the interval $[0, a]$, where a is the maximum estimation for a certain size of *sprint backlog*. We consider that the probability

function of e is the presented below.

$$f(e) = \begin{cases} 0 & \text{si} \quad e \leq 0, \\ \frac{e}{a} & \text{si} \quad e \in [0, a), \\ 1 & \text{si} \quad e \geq a. \end{cases}$$

Since SM is not sure about DT's action at the third stage, SM determines the following expected utility function.

$$E[u_{SM}] = u_{SM}(s, e_H)\, Pr[e_H] + u_{SM}(s, e_{LC})\, Pr[e_{LC}] + u_{SM}(s, e_L)\, Pr[e_L]$$

The expected utility function can be rewritten by replacing the SM's payoff for each effort alternative.

$$E[u_{SM}] = \left((rc)\, e_H - \frac{(rc)^2}{\tau}\right) Pr[e_H] + \left((rc)\, e_{LC} - \frac{(r)^2}{\tau}\right) Pr[e_{LC}] + ((rc)\, e_L)\, Pr[e_L]$$

By proposition 4.1, we know that DT follows ϕ^*_{DT} at the third stage. Therefore, the expected utility function can be expressed as shown below

$$E[u_{SM}] = \left((rc)\, e_H - \frac{(rc)^2}{\tau}\right) Pr[r \geq h_r \text{ and } c \geq h_c] +$$

$$\left((rc)\, e_{LC} - \frac{(r)^2}{\tau}\right) Pr[r \geq h_r \text{ and/or } c < h_c] + ((rc)\, e_L)\, Pr[r < h_r].$$

By substituting the probabilities $f(e)$ for each low-effort alternative and subtracting them from 1 to obtain the high-effort probability, the expected utility function is as follows

$$E[u_{SM}] = \left((rc)\, e_H - \frac{(rc)^2}{\tau}\right)\left(1 - \frac{e_{LC}}{a} - \frac{e_L}{a}\right) + \left((rc)\, e_{LC} - \frac{r^2}{\tau}\right)\left(\frac{e_{LC}}{a}\right) + ((rc)\, e_L)\left(\frac{e_L}{a}\right).$$

According to the decision rule ϕ^*_{DT}, we replace the effort that DT exerts for each effort alternative.

$$E[u_{SM}] = \left((rc)\, \frac{rc}{2\gamma} - \frac{(rc)^2}{\tau}\right)\left(1 - \frac{r+1}{2\gamma a}\right) + \left((rc)\, \frac{r}{2\gamma} - \frac{r^2}{\tau}\right)\left(\frac{r}{2\gamma a}\right) + \left((rc)\, \frac{1}{2\gamma}\right)\left(\frac{1}{2\gamma a}\right)$$

Now, we compute the first-order condition to find the critical points of the expected utility function. Firstly, we look for the SD resources r at equilibrium.

$$\frac{\partial_{E[u_{SM}]}}{\partial_r} = \frac{\tau c + 3r^2(\tau c - 2\gamma) + rc^2(\tau - 2\gamma)(4\gamma a - 3r - 2)}{4\gamma^2 \tau a}$$

There are two critical points of $\frac{\partial_{E[u_{SM}]}}{\partial_r} = 0$.

$$r_1^* = \frac{c^2(\tau-2\gamma)(1-2\gamma a) - \sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2 - 3\tau c\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right)}}{3\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right)}$$

$$r_1^* = \frac{c^2(\tau-2\gamma)(1-2\gamma a) + \sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2 - 3\tau c\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right)}}{3\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right)}$$

Both r_1^* and r_2^* must satisfy that $\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2 < 3\tau c\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right)$ and $(\tau c-2\gamma) - c^2(\tau-2\gamma) \neq 0$.

Then, we compute the second-order condition to verify if r_1^* and r_2^* maximize the SM's payoff.

$$\frac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}} = \frac{3r\left((\tau c-2\gamma) - c^2(\tau-2\gamma)\right) + c^2(\tau-2\gamma)(2\gamma a-1)}{2\gamma^2\tau a}$$

Firstly, by introducing r_1^* into $\frac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}$, we have that

$$\begin{aligned}
&\tfrac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}(r_1^*) \\
&= \frac{3\left(\frac{c^2(\tau-2\gamma)(1-2\gamma a)-\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}}{3\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}\right)\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)+c^2(\tau-2\gamma)(2\gamma a-1)}{2\gamma^2\tau a} \\
&= \frac{c^2(\tau-2\gamma)(1-2\gamma a)-\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}+c^2(\tau-2\gamma)(2\gamma a-1)}{2\gamma^2\tau a} \\
&= -\frac{\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}}{2\gamma^2\tau a}.
\end{aligned}$$

Secondly, by introducing r_2^* into $\frac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}$, we have that

$$\begin{aligned}
&\tfrac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}(r_2^*) \\
&= \frac{3\left(\frac{c^2(\tau-2\gamma)(1-2\gamma a)+\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}}{3\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}\right)\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)+c^2(\tau-2\gamma)(2\gamma a-1)}{2\gamma^2\tau a} \\
&= \frac{c^2(\tau-2\gamma)(1-2\gamma a)+\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}+c^2(\tau-2\gamma)(2\gamma a-1)}{2\gamma^2\tau a} \\
&= \frac{\sqrt{\left(c^2(2\gamma-\tau)(1-2\gamma a)\right)^2-3\tau c\left((\tau c-2\gamma)-c^2(\tau-2\gamma)\right)}}{2\gamma^2\tau a}.
\end{aligned}$$

Since $\frac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}(r_1^*) < 0$ and $\frac{\partial^2_{E[u_{SM}]}}{\partial_{r^2}}(r_2^*) > 0$, we find that r_1^* maximize the SM's payoff and r_2^* minimize the SM's payoff. Therefore, r_1^* is the unique optimal strategy for SM concerning the SD resources to provide to DT. See figure 4.3

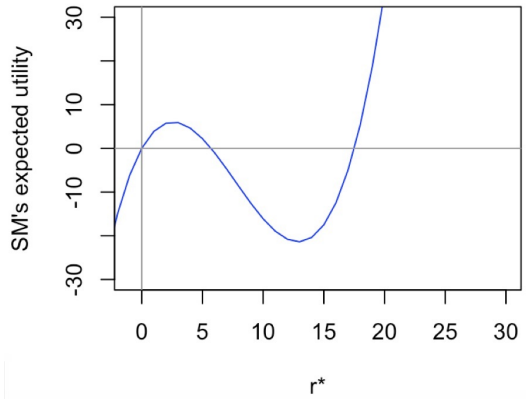

Figure 4.3: Example of *SM* expected utility function.

Source: Own elaboration. Parameters $a = 500, c = 1, \tau = 0.9, \gamma = 0.1$.

Next, we look for the communication investment at equilibrium c.

$$\frac{\partial_{E[u_{SM}]}}{\partial_c} = \frac{r^2 c\,(r+1)}{\gamma\tau a} + \frac{-2r^2 c\,(r+1) + r + r^3}{4\gamma^2 a} + \frac{r^2 c\,(\tau - 2\gamma)}{\gamma\tau}$$

Here, there is one critical point of $\frac{\partial_{E[u_{SM}]}}{\partial_c} = 0$, which is

$$c_1^* = \frac{\tau\left(r^2+1\right)}{2r\,(r+1-2\gamma a)\,(\tau - 2\gamma)}$$

The critical point c_1^* must satisfy that $\tau - 2\gamma \neq 0$, $(r+1) - 2\gamma a \neq 0$, $r^2 + 1 \neq 0$, $\tau \neq 0$ and $r \neq 0$.

Then, we compute the second-order condition to verify if c_1^* maximizes the SM's payoff.

$$\frac{\partial^2_{E[u_{SM}]}}{\partial_{c^2}} = \frac{r^2\,(\tau - 2\gamma)\,(2\gamma a - (r+1))}{2\gamma^2\tau a}$$

Since γ, τ and a are positive parameters, we find that $\frac{\partial^2_{E[u_{SM}]}}{\partial_{c^2}}(c_1^*) < 0$ when $\tau < 2\gamma$ and $2\gamma a > (r+1)$, or when $\tau > 2\gamma$ and $2\gamma a < (r+1)$; that is, c_1^* maximizes the SM's payoff when such conditions are met. Therefore, c_1^* is the unique optimal strategy for SM concerning the communication expense.

Thus, the expected utility function of SM has a unique critical point as declared in proposition 4.2.

Proposition 4.2. *The support vector at equilibrium of the scrum master is unique and it is*

$$s^* = (r^*, c^*) \quad = \quad \left(\frac{c^2(\tau-2\gamma)(1-2\gamma a) - \sqrt{(c^2(2\gamma-\tau)(1-2\gamma a))^2 - 6\tau c((\tau c - 2\gamma) - c^2(\tau-2\gamma))}}{3((\tau c - 2\gamma) - c^2(\tau - 2\gamma))},\ \frac{\tau\left(r^2+1\right)}{2r(r+1-2\gamma a)(\tau-2\gamma)} \right),$$

when $\tau < 2\gamma$ *and* $2\gamma a > (r+1)$, *or when* $\tau > 2\gamma$ *and* $2\gamma a < (r+1)$.

Summarizing the above as a closed solution, the theorem 4.1 states the equilibrium strategies of SM and DT in the offshore EE game.

Theorem 4.1. *The offshore EE game has a unique Bayesian sub-game perfect Nash equilibrium* (s^*, ϕ^*)*, where*

$$\phi^*_{ED}(s,h) = \begin{cases} e_H & si \quad r \geq h_r \text{ and } c \geq h_c, \\ e_{LC} & si \quad r \geq h_r \text{ and } c < h_c, \\ e_L & si \quad r < h_r, \end{cases}$$

and

$$s^* = (r^*, c^*) \quad = \quad \left(\frac{c^2(\tau-2\gamma)(1-2\gamma a) - \sqrt{(c^2(2\gamma-\tau)(1-2\gamma a))^2 - 6\tau c((\tau c-2\gamma) - c^2(\tau-2\gamma))}}{3((\tau c-2\gamma) - c^2(\tau-2\gamma))}, \frac{\tau\left(r^2+1\right)}{2r(r+1-2\gamma a)(\tau-2\gamma)} \right),$$

when $\tau < 2\gamma$ *and* $2\gamma a > (r+1)$*, or when* $\tau > 2\gamma$ *and* $2\gamma a < (r+1)$*.*

4.4.3 Analysis of equilibria

This section analyzes how equilibrium strategies and payoffs change as exogenous variables change in a GSD context.

Proposition 4.3. ***Comparative statics in the offshore EE game.*** *Consider the Bayesian equilibrium* (s^*, ϕ^*) *of the offshore EE game in Theorem 4.1. The following relationships hold for the parameters* τ *and* γ*.*

(a) $\frac{\partial c^*}{\partial \tau} > 0$ *when* $r + 1 < 2\gamma a$*,*

(b) $\frac{\partial c^*}{\partial \gamma} > 0$ *when* $\tau a + r + 1 > 4\gamma a$*,*

(c) $\frac{\partial_{r^*}}{\partial_{1-2\gamma a}} > 0$ *when* $\tau c - 2\gamma < (\tau - 2\gamma)\, c^2$*,*

(d) $\frac{\partial_{r^*}}{\partial_{\tau-2\gamma}} > 0$ *when* $\tau c - 2\gamma > (\tau - 2\gamma)\, \frac{c}{3\tau} \left((1-2\gamma a)^2 (\tau c - 2\gamma) + 3\tau c \right)$*, and*

(e) $\frac{\partial_{r^*}}{\partial_{\tau c-2\gamma}} > 0$ *when* $\tau c - 2\gamma < (\tau - 2\gamma)\, \frac{c}{3\tau} \left((1-2\gamma a)^2 (\tau - 2\gamma) c^2 + 3\tau c \right)$*.*

Proof. See Appendix B. □

Proposition 4.3 shows that the relationship between equilibrium strategies and the exogenous parameters may be either negative or positive, depending on certain conditions. Concerning communication

investment, it increases as the environmental stability τ increases when $r+1 < 2\gamma a$; on the contrary, the relationship will be negative. That is, the communication strategy must increase as the environmental stability increases and the SD resources are insufficient to deal twice the complexity of the *sprint backlog*. Also, the relationship between the *sprint backlog* complexity γ and communication investment could change, it is positive when $\tau a + r + 1 > 4\gamma a$ and negative on the contrary. In this case, the comunication strategy must increase as the *sprint backlog* complexity increases as long as the environmental stability as well as the SD resources are enough to cope with four times the complexity. In other words, the mix of the the current environmental situation together with the SD resources that are available to face the *sprint backlog* complexity define whether the comunication investment strategy should be greater as the exogenous factors increase.

Regarding to SD resources provision, the relationship with the exogenous parameters is more complex. We analyze how it behaves as its terms change. For example, the SD resources provision increases as the difference $1 - 2\gamma a$ increases when $\tau c - 2\gamma$ is below $(\tau - 2\gamma)\, c^2$. Concerning the increased difference between stability and twice the complexity $\tau - 2\gamma$, we find that such difference adjusted by $\frac{c}{3\tau}\left((1-2\gamma a)^2 (\tau c - 2\gamma) + 3\tau c\right)$ must be below $\tau c - 2\gamma$, in order to increase the SD resources provision. On the contrary, the previous difference $\tau - 2\gamma$ adjusted by $\frac{c}{3\tau}\left((1-2\gamma a)^2 (\tau - 2\gamma)\, c^2 + 3\tau c\right)$ must be up to $\tau c - 2\gamma$, in order to increase the SD resources provision as $\tau c - 2\gamma$ increases.

Now, we show the relationship between the SM expected payoff and the exogenous parameters.

Proposition 4.4. ***Changes in*** SM ***expected payoff.*** *Consider the SM's expected utility function* $E[u_{SM}]$. *The following relationships hold for the parameters* τ *and* γ.

(a) $\frac{\partial_{E[u_{SM}]}}{\partial_\tau} > 0$ *when* $r > c^2(r + 1 - 2\gamma a)$,

(b) $\frac{\partial_{E[u_{SM}]}}{\partial_\gamma} > 0$ *when* $r\gamma\left(r^2 - c^2(\tau a + r + 1)\right) > c\tau\left(1 + r^2 - rc(r+1)\right)$.

Proof. See Appendix B. □

Proposition 4.4 shows that the relationship between the SM's expected payoff and the enviromental stability is positive, when $r > c^2(r + 1 - 2\gamma a)$. In other words, under more stable conditions, the SM expected payoff increases as long as the SD resources and their ratio to the communication

resources squared are enough to handle twice the *sprint backlog* complexity. There is also a positive relationship between the SM's expected payoff and the *sprint backlog* complexity. Here, the product of the SD resources times the *spring backlog* complexity fitted by the term $r^2 - c^2(\tau a + r + 1)$ must be greater than the product of the communication resources times the environmental stability fitted by $1 + r^2 - rc(r + 1)$. Thus, when complexity is greater, the SM's expected payoff can only increase when the appropiate relationship of strategies is applied in the provision of development resources to manage the complexity as well as communication resources to face the instability issues.

We carry out some numerical examples in the next section to show the previous relationships.

4.5 Numerical examples

In this section we perform numerical examples to show the changes of equilibrium strategies (s^*, ϕ^*) and payoffs u_{SM} and u_{DT} as exogenous variables change. We consider positive values of τ and γ, since they range as $0.43 \geq \tau \geq 1$ and $0 > \gamma \geq 1$ (Ziauddin and Zia, 2012).

By simplicity, we are considering that $I = 0$. Table 4.1 shows the values for the τ, γ and a parameters.

Table 4.1: Values assigned to exogenous variables

Parameters	Values	Interpretation
τ	0.43, 0.65, 0.9	Low, Medium and High environmental stability
γ	0.1, 0.5	Low and Medium sprint backlog complexity
a	10	Maximum effort estimation size of sprint backlog

Source: Own elaboration.

In order to compute the equilibrium estrategies (s^*, ϕ^*), we begin by setting two different quantities of communication investment c^*. See table 4.2 and figure 4.4.

Considering what propositions 4.1, 4.2, 4.3 and 4.4 state, we observe the following behavior in players' strategies and payoff. There is a negative relationship between the environmental stability τ and the strategies of SD resources provision r^* and DT's effort ϕ^* when the *sprint backlog* complexity γ

Table 4.2: Equilibrium strategies and payoffs as τ and γ change with different initial amounts of communication investment

		scrum master			offshore team	
τ	γ	r^*	c^*	$E[u_{SM}]$	ϕ^*	u_{DT}
0.43	0.1	0.9589	8	39.0209	38.3591	147.1424
0.43	0.5	0.0052	8	0.0020	0.0419	0.0008
0.65	0.1	0.8886	8	41.0087	35.5465	126.356
0.65	0.5	0.0129	8	0.0051	0.1032	0.0053
0.9	0.1	0.8636	8	42.4551	34.5450	119.336
0.9	0.5	0.0638	8	0.0253	0.5106	0.1304
0.43	0.1	2.7586	2	22.7639	27.5864	76.1012
0.43	0.5	0.0210	2	0.0021	0.0420	0.0008
0.65	0.1	2.0206	2	16.1578	20.2064	40.8301
0.65	0.5	0.0521	2	0.0051	0.1042	0.0054
0.9	0.1	1.8289	2	14.7687	18.2894	33.4503
0.9	0.5	0.2928	2	0.0276	0.5856	0.1714

Source: Own elaboration. Parameters $I = 0, a = 10$.

is low. This relationship becomes positive when γ is at medium level. Meanwhile, the relationship between the *sprint backlog* complexity and the strategies r^* and ϕ^* is always negative. Concerning the players' payoff, they follow the same behavior as the players' strategies. The exception emerges when the communication strategy c^* is too high and the *sprint backlog* complexity is low; in this case, the SM payoff increases as the environmental stability increases. In other words, more complexity always lowers players' strategies and profits while more stability mixed with medium complexity boost them. However, more stability mixed with low complexity decrease players' strategies and payoffs; unless the communication investment strategy c^* is too high, which produces higher benefits for SM.

Figure 4.4 plots all these relations; it can be seen how the players' strategies get restrictive while

the players' payoff grows as the environment becomes more stable when γ is low; also, the opposite behavior is presented when γ is at medium level.

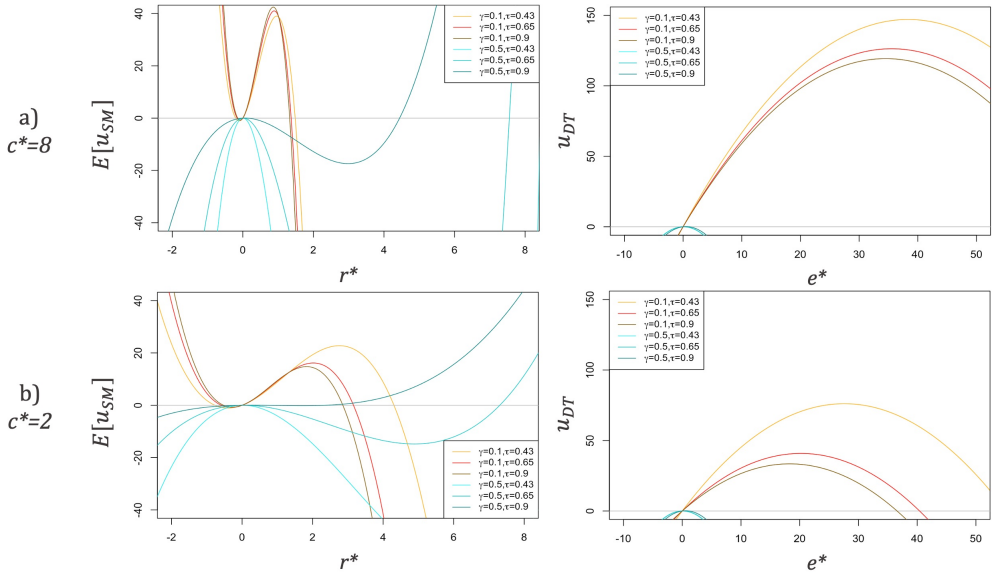

Figure 4.4: Changes in equilibrium strategies and payoffs as τ and γ change with different initial amounts of communication investment.

Source: Own elaboration. Parameters $I = 0, a = 10$.

Now, we set two different quantities of SD resources r^* to compute the equilibium strategies and payoffs. See table 4.3 and figure 4.5.

Concerning the environmental stability τ, we observe the same behavior as the previous example. That is, there is a negative relationship between τ and the strategies of communication resources provision c^* and DT's effort ϕ^* when the *sprint backlog* complexity γ is low. Also, this relationship becomes positive when γ is at medium level. As well, the players' payoff follow the same behavior as the players' strategies. As before, the same exception emerges when the SD resources strategy r^* is too high and the *sprint backlog* complexity is low which produces the SM payoff increases as the environmental stability increases.

In contrast, the relationship between the *sprint backlog* complexity and players' strategies (c^*, ϕ^*) and their payoffs is somewhat chaotic as long as the SD resources strategy r^* is high. Note that players'

Table 4.3: Equilibrium strategies and payoffs as τ and γ change with different initial amounts of SD resources

		scrum master			offshore team	
τ	γ	r^*	c^*	$E[u_{SM}]$	ϕ^*	u_{DT}
0.43	0.1	8	1.0850	109.9113	43.4006	188.3614
0.43	0.5	8	3.0646	-39.3877	24.5175	300.555
0.65	0.1	8	0.8382	151.0447	33.5317	112.4378
0.65	0.5	8	7.5446	117.3914	60.3571	1821.492
0.9	0.1	8	0.7461	200.5683	29.8469	89.0839
0.9	0.5	8	36.5625	893.7361	292.5	42778.12
0.43	0.1	2	2.3369	19.9096	23.3695	54.6136
0.43	0.5	2	0.1347	-1.7931	0.2694	0.0362
0.65	0.1	2	1.8055	16.4156	18.0555	32.6003
0.65	0.5	2	0.3316	-1.0649	0.6632	0.2199
0.9	0.1	2	1.6071	15.6448	16.071	25.8290
0.9	0.5	2	1.6071	-0.0853	3.2142	5.1658

Source: Own elaboration. Parameters $I = 0, a = 10$.

strategies and payoff follow the same behavior as the previous example when the SD resources are low. However, when r^* is high and γ increases, the strategies (c^*, ϕ^*) also increase unless τ is low; meanwhile, SM's payoff decreases unless τ is high and the DT's payoff increases regardless of τ levels. It means that the effort increases when support strategies (r^*, c^*) are enough to cope with the combination of stability and complexity levels; on the contrary, the effort strategy decreases. It is worth noting that here, high stability generates big wins for both players, even if complexity gets higher. Figure 4.5 traces all these behaviors.

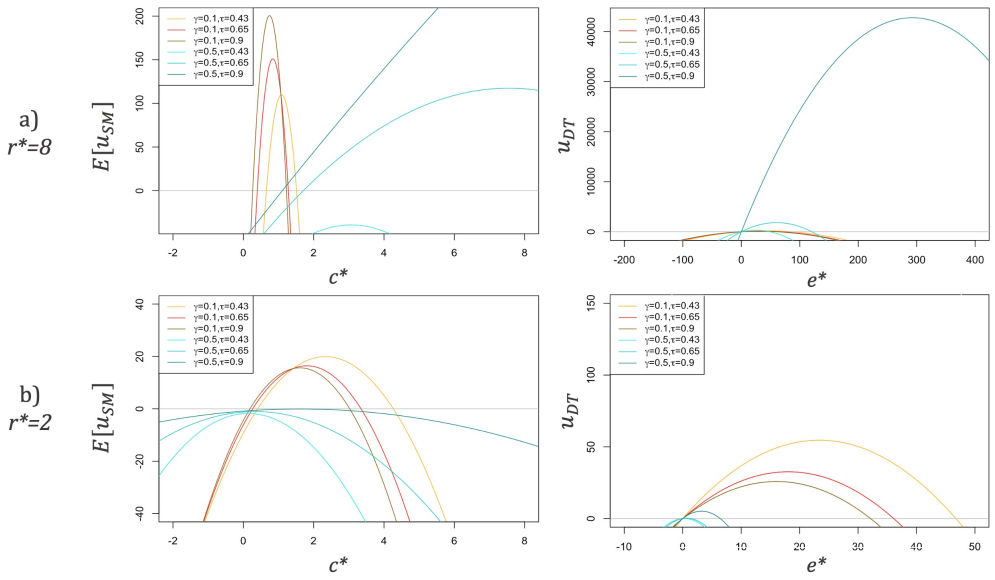

Figure 4.5: Changes in equilibrium strategies and payoffs as τ and γ change with different initial amounts of SD resources.

Source: Own elaboration. Parameters $I = 0, a = 10$.

4.6 Conclusions

In this research, we study the impact of leader-team interaction on effort estimation and support provision during Agile Global Software Development (AGSD). We model such interaction using a sequential game-theoretical approach to analyze players' strategies in the presence of incomplete information. As the scrum master has limited information about the skills of the team located at an offshore site, principal-agent issues arise during the EE process. Findings are important for those companies that move software development to a branch or to an external third party; that is, companies developing software under offshoring or outsourcing arrangements.

Based on the approach of agile SD, the scrum master has the responsability to support the team by removing obstacles. We investigate the magnitude of support provided and the effort estimations at equilibrium. The support basket in AGSD includes SD resources and also, investment in communication to overcome problems due to distance between sites collaborating on a project. Therefore, we focus on analyzing players' strategies (support and effort) at equilibrium during a single sprint where the

scrum master is at onsite and the development team is located offshore. We model such interaction as an strategic model which we name as the offshore EE game where the scrum master has incomplete information about team's abilities. Here, nature plays first determining the type of team; thus, players make decisions strategically to maximize their profits. Then, the scrum master acts supporting with an expectation about her benefits since she does not know team's type; By last, the team observes the support and chooses the effort level for the sprint. We find the solution by computing the Bayesian sub-game perfect Nash equilibrium and using the players' payoff functions described in section 3.4.1 for the EE game model, just without taking players' types into account. We also analyze how behave players' strategies and payoff as the exogenous variables change in a GSD environment.

Results show that team's optimal strategy is unique at the third stage of the game. Then, a high effort level come about when both SD and communication resources are enough to deal with the *sprint backlog* complexity based on team's experience. On the contrary, effort decreases either in communication or in development as a whole, depending on which resources the team considers useless. At second stage of the game, we find that the scrum master's optimal strategy is also unique and consists of a support vector that provides SD and communication resources. However, the scrum master's strategies are highly dependent on the availability of a database with reliable effort records. Theorem 4.1 show players' strategies at equilibrium in the offshore EE game.

By analyzing Bayesian equilibria, we find that the relationship between players' strategies and payoff, and the exogenous variables can be positive or negative. It depends on whether the SD resources are sufficient to face with the complexity despite the environmental situation; as well as whether the communication resources together with current environment allows to deal the complexity of development.

Numerical examples show that complexity and stability levels impact equilibrium strategies and payoffs amounts in a different way under GSD environments. A low complexity produces a reduction in equilibrium strategies, as well as an increase in players' payoff as the stability increases. On the contrary, this behavior is reversed when complexity goes to medium levels. However, complexity tends to bring down players' strategies and profits; especially, the stability together with low complexity diminish players' strategies and payoff. Interestingly, the combination of stability and medium

complexity increases them. It is worth noting that high stability boost players' benefits when SD resources are considerable, even if complexity levels increase.

Like the previous EE model in chapter 3, we analyze effort estimation and support provision within an iteration. We pretend to extend this offshore EE game theoretical model to a dynamic framework that allow to analyze EE and support provision over several sprints by updating iteration velocity. As well, we see as another future work to study how interaction between players around scrum meetings impacts the offshore EE game's outcomes.

Chapter 5

Conclusions

5.1 Conclusions

As digitalization expands to an increasing number of economic and social activities, the prevailing need for success in the development of software projects highlights. Here, effort estimation (EE) is a fundamental process for software development (SD) planning. Agile methodologies perform EE at the beginning of each iteration by considering changes' impact during development. So, EE becomes central to achieve sucessful results in software projects delivery. In chapter 2, we describe top-down and bottom-up EE strategies that use diverse available techniques and methodologies to estimate effort. However, there is a mutual depdendency concerning both strategies during the EE process in an agile environment. In a Scrum framework, the scrum master and the development team interact in a top-down or bottom-up relationship through Scrum ceremonies. Both parties behave strategically as they pursue different goals while they cope with environmental and dificulty factors. In this thesis, we address EE as an strategic model by following a game-theoretical approach. This is the main contribution of the present research. The strategic analysis of EE allows to find the optimal solution to maximize every iteration's results. Thus, determining EE considering the parties' strategic behavior contributes to innovation in software projects management.

We rely on the principal-agent theory to study, firstly, the complete information case where the parties

know each other and, secondly, the incomplete information case that deals with uncertainty about the abilities of team members.

Regarding the complete information case, we develop a sequential game-theoretic model of EE where the parties involved know each other's skills. Therefore, this EE game model focuses on insourced SD with mature teams, where the scrum master provides resources to the team while the team exerts effort. Computing the sub-game perfect Nash equilibrium, there is a unique optimal solution for both players when the iteration ends. Interestingly, we find that the structure of leader-team interaction does not change EE; while players' benefits as well as the leader's strategy do. Furthermore, multiple equilibria arise when the scrum master acts before and after the team. Then, the leadership style plays an important role in achieving the goals of the iteration. Nevertheless, exogenous factors (stability, difficulty, experience) always impact on players' strategies and payoffs; although it is worth mentioning, the team estimates effort by compensating for their lack of experience and skills with the leader's expertise, and vice versa.

Now, the incomplete information case concerns companies that move SD to a branch or that outsource SD. Specifically, we focus on offshoring squemes where the scrum master has limited information about team's skills since he is located at offshore site. Therefore, we model the offshoring EE game to deal with uncertainties about team' type. Due to the physical distance between the parties, the scrum master provides communication support in addition to SD resources, while the team strives. Finding the Bayesian sub-game perfect Nash equilibrium, there is also a unique optimal solution that behaves inversely to the complete information case for certain combinations of exogenous factors (stability and complexity). In particular, as stability increases, medium levels of complexity increase players' strategies and payoffs; while low complexity decreases them. It is worth noting that high stability boosts players' strategies and profits, even when the complexity goes up.

Scrum is a methodology to deal with the change impact due to issues related with people, technologies, installations, information, dependencies and others. So, both EE models include contextual factors of agile SD environments, such as environmental stability and *sprint backlog* complexity. Also, they use quadratic costs as well as Cobb-douglas production functions to compute player payoffs. In particular, the EE offshore model depends on historical data base of EE in order to calculate the scrum master's

expected utility. Therefore, the resulting optimal strategies are expected to be succesful in achieving the sprint's goals as more accurate be the contextual factors' measurements and EE historical records. However, both EE models embrace the impact of interactions between agents during the EE process in agile environments, which adds strategic precision on estimations of effort and resources.

Finally, concerning the hypotheses of this research, we found that a strategic analysis of EE considering contextual factors in agile SD allows to determine the optimal strategies at equilibrium to deliver the customer's requeriments on time. Mainly, we found that the leadership type of scrum master is core to outcomes, as multiple equilibria arise after a retrospective meeting take place.

5.2 Future work

It is worth emphasizing that our approach deals with the *sprint backlog* complexity and environmental stability during software development inside an iteration. So, in futures works, we pretend to extend our game-theoretical model to dynamic frameworks where it is important to differentiate between user stories as well as updating iteration velocity.

Given that interactions are getting more complex in global SD environments, we intend to study in a future work, how players interaction around scrum meetings impacts the offshore EE game's results. Furthermore, in a broad approach, we consider combining both EE models to generate a distributed EE model involving several team members located at different sites.

By last, it would be useful future work to test the EE game models in an agile SD company that allows feeding the models with their environment and complexity indicators, as well as their historical EE records. It might reveal interesting findings about the EE models performance in the empirical field.

Appendix

Appendix A

Proposition 3.4

Proof. By Proposition 3.1, in the original game, the SM and DT equilibrium strategies are

$$r^* = \frac{9\ \tau^2 h f^2}{32\ \gamma} - K \quad \text{and} \quad \epsilon^* = \frac{9\ \tau^2 h^2 f^2}{64\ \gamma^2} - G,$$

respectively.

To analyze the relationship between the equilibrium strategies and the model's parameters (f, h, τ, γ), we compute the partial derivative of r^* and ϵ^* with respect to such exogenous variables. We have that

$$\begin{aligned}
\frac{\partial r^*}{\partial f} &= \frac{9\ \tau^2 h f}{16\ \gamma}, & \frac{\partial \epsilon^*}{\partial f} &= \frac{9\ \tau^2 h^2 f}{32\ \gamma^2},\\
\frac{\partial r^*}{\partial \tau} &= \frac{9\ \tau h f^2}{16\ \gamma}, & \frac{\partial \epsilon^*}{\partial \tau} &= \frac{9\ \tau h^2 f^2}{32\ \gamma^2},\\
\frac{\partial r^*}{\partial h} &= \frac{9\ \tau^2 f^2}{32\ \gamma}, & \frac{\partial \epsilon^*}{\partial h} &= \frac{9\ \tau^2 h f^2}{32\ \gamma^2},\\
\frac{\partial r^*}{\partial \gamma} &= -\frac{9\ \tau^2 h f^2}{32\ \gamma^2}, & \frac{\partial \epsilon^*}{\partial \gamma} &= -\frac{9\ \tau^2 h^2 f^2}{32\ \gamma^3}.
\end{aligned}$$

Since f, τ, h, and γ are positive, it is clear that

- $\frac{\partial r^*}{\partial f} > 0$, $\frac{\partial \epsilon^*}{\partial f} > 0$,
- $\frac{\partial r^*}{\partial \tau} > 0$, $\frac{\partial \epsilon^*}{\partial \tau} > 0$,
- $\frac{\partial r^*}{\partial h} > 0$, $\frac{\partial \epsilon^*}{\partial h} > 0$, and

- $\frac{\partial r^*}{\partial \gamma} < 0,\ \frac{\partial \epsilon^*}{\partial \gamma} < 0.$

□

Proposition 3.5

Proof. By Proposition 3.2, the players' equilibrium strategies at scenario 1 are

$$r^{*2} = \frac{3\ \tau^2 h f^2}{16\ \gamma} - K, \qquad \epsilon^{*2} = \frac{9\ \tau^2 h^2 f^2}{64\ \gamma^2} - G.$$

We analyze the relationship between the additional resources provided by SM, the effort set by DT, and the parameters f, τ, h, γ. The partial derivatives are

$$\begin{aligned}
\frac{\partial r^{*2}}{\partial f} &= \frac{3\ \tau^2 h f}{8\ \gamma}, & \frac{\partial \epsilon^{*2}}{\partial f} &= \frac{9\ \tau^2 h^2 f}{32\ \gamma^2},\\
\frac{\partial r^{*2}}{\partial \tau} &= \frac{3\ \tau h f^2}{8\ \gamma}, & \frac{\partial \epsilon^{*2}}{\partial \tau} &= \frac{9\ \tau h^2 f^2}{32\ \gamma^2},\\
\frac{\partial r^{*2}}{\partial h} &= \frac{3\ \tau^2 f^2}{16\ \gamma}, & \frac{\partial \epsilon^{*2}}{\partial h} &= \frac{9\ \tau^2 h f^2}{32\ \gamma^2},\\
\frac{\partial r^{*2}}{\partial \gamma} &= -\frac{3\ \tau^2 h f^2}{16\ \gamma^2}, & \frac{\partial \epsilon^{*2}}{\partial \gamma} &= -\frac{9\ \tau^2 h^2 f^2}{32\ \gamma^3}.
\end{aligned}$$

Since f, τ, h, and γ must be positive, we prove that

- $\frac{\partial_{r^{*2}}}{\partial_f} > 0,\ \frac{\partial_{\epsilon^{*2}}}{\partial_f} > 0,$
- $\frac{\partial_{r^{*2}}}{\partial_\tau} > 0,\ \frac{\partial_{\epsilon^{*2}}}{\partial_\tau} > 0,$
- $\frac{\partial_{r^{*2}}}{\partial_h} > 0,\ \frac{\partial_{\epsilon^{*2}}}{\partial_h} > 0,$ and
- $\frac{\partial_{r^{*2}}}{\partial_\gamma} < 0,\ \frac{\partial_{\epsilon^{*2}}}{\partial_\gamma} < 0.$

□

Proposition 3.6

Proof. Proposition 3.3 establishes the players' equilibrium strategies at scenario 2, which are

$$0 \le r_1^* \le \frac{3\ \tau^2 h f^2}{16\ \gamma} - K,\ r_2^* = \frac{3\ \tau^2 h f^2}{16\ \gamma} - K - r_1 \qquad \epsilon^{*3} = \frac{9\ \tau^2 h^2 f^2}{64\ \gamma^2} - G.$$

We analyze the relationship between the additional resources provided by SM, the effort set by DT, and the parameters f, τ, h, γ. The partial derivatives are

$$\begin{aligned}
\frac{\partial r_2^*}{\partial f} &= \frac{3\ \tau^2 h f}{8\ \gamma}, & \frac{\partial \epsilon^{*3}}{\partial f} &= \frac{9\ \tau^2 h^2 f}{32\ \gamma^2},\\
\frac{\partial r_2^*}{\partial \tau} &= \frac{3\ \tau h f^2}{8\ \gamma}, & \frac{\partial \epsilon^{*3}}{\partial \tau} &= \frac{9\ \tau h^2 f^2}{32\ \gamma^2},\\
\frac{\partial r_2^*}{\partial h} &= \frac{3\ \tau^2 f^2}{16\ \gamma}, & \frac{\partial \epsilon^{*3}}{\partial h} &= \frac{9\ \tau^2 h f^2}{32\ \gamma^2},\\
\frac{\partial r_2^*}{\partial \gamma} &= -\frac{3\ \tau^2 h f^2}{16\ \gamma^2}, & \frac{\partial \epsilon^{*3}}{\partial \gamma} &= -\frac{9\ \tau^2 h^2 f^2}{32\ \gamma^3}.
\end{aligned}$$

Since f, τ, h, and γ must be positive, we prove that

- $\frac{\partial r_1^*}{\partial f} = 0$, $\frac{\partial r_2^*}{\partial f}$, $\frac{\partial e^{*3}}{\partial f} > 0$,
- $\frac{\partial r_1^*}{\partial \tau} = 0$, $\frac{\partial r_2^*}{\partial \tau}$, $\frac{\partial e^{*3}}{\partial \tau} > 0$,
- $\frac{\partial r_1^*}{\partial h} = 0$, $\frac{\partial r_2^*}{\partial h}$, $\frac{\partial e^{*3}}{\partial h} > 0$, and
- $\frac{\partial r_1^*}{\partial \gamma} = 0$, $\frac{\partial r_2^*}{\partial \gamma}$, $\frac{\partial e^{*3}}{\partial \gamma} < 0$.

□

Proposition 3.7

Proof. We compare the equilibrium effort strategy between the three versions of the EE game. From expressions (4), (7) and (9), we know that

$$\begin{aligned}
\epsilon^* &= \frac{(K+r)\ h}{2\ \gamma} - G,\\
\epsilon^{*2} &= \frac{9\ h^2\ f^2\ \tau^2}{64\ \gamma^2} - G,\\
\epsilon^{*3} &= \frac{9\ h^2\ f^2\ \tau^2}{64\ \gamma^2} - G.
\end{aligned}$$

By introducing r^* into ϵ^*, we find the equilibrium effort strategy in the original EE game:

$$\epsilon^* = \frac{(K + \frac{9\ \tau^2\ h\ f^2}{32\ \gamma} - K)\ h}{2\ \gamma} - G = \frac{9\ \tau^2\ h^2\ f^2}{64\ \gamma^2} - G.$$

Therefore, we have that $\epsilon^* = \epsilon^{*2} = \epsilon^{*3}$. □

Appendix B

Proposition 4.3

Proof. By theorem 4.1, we know the players' equilibrium strategies in the offshore EE game. We analyze the relationship between the SM's equilibrium strategies and the parameters τ and γ by computing the partial derivatives.

Firstly, we analyze the communication investment strategy

$$c^* = \frac{\tau\left(r^2+1\right)}{2r\left(r+1-2\gamma a\right)\left(\tau-2\gamma\right)}.$$

- The partial derivative in relation with the environmental stability is

$$\frac{\partial c^*}{\partial \tau} = -\frac{\gamma\left(r^2+1\right)}{r\left(r+1-2\gamma a\right)\left(\tau-2\gamma\right)^2}.$$

  Since τ and r must be positive, and quadratic terms do not modify the sign of the partial derivative, it is negative when $r+1>2\gamma a$ and positive when $r+1<2\gamma a$. Therefore, we prove that

  $\frac{\partial c^*}{\partial \tau} < 0$ when $r+1>2\gamma a$, and

  $\frac{\partial c^*}{\partial \tau} > 0$ when $r+1<2\gamma a$.

- The partial derivative in relation with the sprint backlog complexity is

$$\frac{\partial c^*}{\partial \gamma} = \frac{\tau\left(r^2+1\right)\left(\tau a+r+1-4\gamma a\right)}{r\left(r+1-2\gamma a\right)^2\left(\tau-2\gamma\right)^2}.$$

  Since τ and r must be positive, and quadratic terms do not modify the sign of the partial derivative, it is negative when $\tau a+r+1<4\gamma a$ and positive when $\tau a+r+1>4\gamma a$. Therefore, we prove that

  $\frac{\partial c^*}{\partial \gamma} < 0$ when $\tau a+r+1<4\gamma a$, and

  $\frac{\partial c^*}{\partial \gamma} > 0$ when $\tau a+r+1>4\gamma a$.

Secondly, we analyze the SD resources provision strategy

$$r^* = \frac{c^2\left(\tau-2\gamma\right)\left(1-2\gamma a\right)-\sqrt{\left(c^2\left(2\gamma-\tau\right)\left(1-2\gamma a\right)\right)^2-6\tau c\left(\left(\tau c-2\gamma\right)-c^2\left(\tau-2\gamma\right)\right)}}{3\left(\left(\tau c-2\gamma\right)-c^2\left(\tau-2\gamma\right)\right)},$$

By making the following changes $x = (\tau - 2\gamma)$, $y = (1 - 2\gamma a)$ and $z = (\tau c - 2\gamma)$, we rewrite the SD resources provision strategy as

$$r^* = \frac{c^2 xy - \sqrt{(c^2 xy)^2 - 6\tau c\,(z - c^2 x)}}{3\,(z - c^2 x)},$$

The partial derivatives in relation with the terms $\tau - 2\gamma$, $1 - 2\gamma a$, and $\tau c - 2\gamma$, are

$$\frac{\partial_{r^*}}{\partial_x} = c^2 y - \frac{c^4 xy^2 z + 3c^5 x\tau - 3c^3 \tau z}{3\,(z - c^2 x)^2 \sqrt{c^4 x^2 y^2 - 6\tau c(z - c^2 x)}},$$

$$\frac{\partial_{r^*}}{\partial_y} = c^2 x - \frac{c^4 x^2 y}{3\,(z - c^2 x)\sqrt{c^4 x^2 y^2 - 6\tau c(z - c^2 x)}},$$

$$\frac{\partial_{r^*}}{\partial_z} = \frac{c^4 x^2 y^2 + 3c^3 x\tau - 3c\tau z}{3\,(z - c^2 x)^2 \sqrt{c^4 x^2 y^2 - 6\tau c(z - c^2 x)}}.$$

Since we have only positive parameters and considering that quadratic terms do not modify the sign of the previous derivatives, they are positive when

- Concerning the term $\tau - 2\gamma$, the partial derivative is positive when $3\tau z > xc\left(y^2 z + 3\tau c\right)$,
- Concerning the term $1 - 2\gamma a$, the partial derivative is positive when $z < xc^2$,
- Concerning the term $\tau c - 2\gamma$, the partial derivative is positive when $3\tau z < xc\left(y^2 x c^2 + 3\tau c\right)$.

Therefore, we prove that

$\frac{\partial_{r^*}}{\partial_{1-2\gamma a}} > 0$ when $\tau c - 2\gamma < (\tau - 2\gamma)\,c^2$, and

$\frac{\partial_{r^*}}{\partial_{\tau - 2\gamma}} > 0$ when $\tau c - 2\gamma > (\tau - 2\gamma)\,\frac{c}{3\tau}\left((1 - 2\gamma a)^2\,(\tau c - 2\gamma) + 3\tau c\right)$, and

$\frac{\partial_{r^*}}{\partial_{\tau c - 2\gamma}} > 0$ when $\tau c - 2\gamma < (\tau - 2\gamma)\,\frac{c}{3\tau}\left((1 - 2\gamma a)^2\,(\tau - 2\gamma)\,c^2 + 3\tau c\right)$. □

Proposition 4.4

Proof. We analyze how the parameters τ and γ change the SM's payoff. We compute the partial derivatives of the SM's expected utility function

$$E\left[u_{SM}\right] = \left((rc)\,\frac{rc}{2\gamma} - \frac{(rc)^2}{\tau}\right)\left(1 - \frac{r+1}{2\gamma a}\right) + \left((rc)\,\frac{r}{2\gamma} - \frac{r^2}{\tau}\right)\left(\frac{r}{2\gamma a}\right) + \left((rc)\,\frac{1}{2\gamma}\right)\left(\frac{1}{2\gamma a}\right).$$

- The partial derivative in relation with the environmental stability is

$$\frac{\partial_{E[u_{SM}]}}{\partial_\tau} = \frac{r^2c^2\left(2\gamma a - r - 1\right) + r^3}{2\gamma a\tau^2} = \frac{r^2\left(2\gamma ac^2 + r - c^2(r+1)\right)}{2\gamma a\tau^2}.$$

  Since r and c, as well as γ, τ and a are positive parameters, the SM's payoff is positive when

$$\begin{aligned} 2\gamma ac^2 + r &> c^2(r+1) \\ r &> c^2(r+1) - 2\gamma ac^2 \\ r &> c^2(r+1-2\gamma a) \end{aligned}$$

  Therefore, we prove that $\frac{\partial_{E[u_{SM}]}}{\partial_\tau} > 0$ when $r > c^2(r+1-2\gamma a)$.

- The partial derivative in relation with the sprint backlog complexity is

$$\frac{\partial_{E[u_{SM}]}}{\partial_\gamma} = \frac{r\left(rc^2\tau\left(r+1\right) + r^3\gamma - rc^2\gamma\left(\tau a + r + 1\right) + c\tau\left(r^2+1\right)\right)}{2\gamma^3\tau a}$$

  .

  Since r and c, as well as γ, τ and a are positive parameters, the SM's payoff is positive when

$$\begin{aligned} rc^2\tau\left(r+1\right) + r^3\gamma &> rc^2\gamma\left(\tau a + r + 1\right) + c\tau\left(r^2+1\right) \\ r^3\gamma &> rc^2\gamma\left(\tau a + r + 1\right) + c\tau\left(r^2+1\right) - rc^2\tau\left(r+1\right) \\ r^3\gamma &> rc^2\gamma\left(\tau a + r + 1\right) + c\tau\left(1 + r^2 - rc\left(r+1\right)\right) \\ r^3\gamma - rc^2\gamma\left(\tau a + r + 1\right) &> c\tau\left(1 + r^2 - rc\left(r+1\right)\right) \\ r\gamma\left(r^2 - c^2\left(\tau a + r + 1\right)\right) &> c\tau\left(1 + r^2 - rc\left(r+1\right)\right) \end{aligned}$$

  Therefore, we prove that $\frac{\partial_{E[u_{SM}]}}{\partial_\gamma} > 0$ when $r\gamma\left(r^2 - c^2\left(\tau a + r + 1\right)\right) > c\tau\left(1 + r^2 - rc\left(r+1\right)\right)$.

□

Bibliography

Acevedo Borrego, A., Linares Barrantes, C., and Cachay Boza, O. (2010). Modelo de análisis y formulación estratégica. empleando herramientas matriciales. *Industrial Data*, 13(1):9–17.

Alzoubi, Y. I. and Gill, A. Q. (2014). Agile global software development communication challenges: A systematic review. In *Pacific Asia Conference on Information Systems (PACIS) Proceedings*.

Amankwah-Amoah, J., Khan, Z., Wood, G., and Knight, G. (2021). Covid-19 and digitalization: The great acceleration. *Journal of Business Research*, 136:602–611.

Araújo, R. d. A., Oliveira, A. L. I., and Meira, S. (2017). A class of hybrid multilayer perceptrons for software development effort estimation problems. *Expert Systems With Applications*, 90:1–12.

Arias, G., Vilches, D. Banchoff, C., Harari, I., Harari, V., and Juliano, P. (2012). The 7 key factors to get successful results in the it development projects. *Procedia Technology*, 5:199–207.

Austin, R. D. (2001). The effects on time pressure on quality in software development: an agency model. *Information Systems Research*, 12:195–207.

Azanha, A., Argoud, A., Camargo Junior, J., and Antoniolli, P. (2017). Agile project management with scrum: A case study of a brazilian pharmaceutical company it project. *International Journal of Managing Projects in Business*, 10(1):121–142.

Azzeh, M. and Nassif, A. B. (2017). Analyzing the relationship between project productivity and environment factors in the case points method. *Journal of Software: Evolution and Process*, 29:1–19.

Bacon, D. F., Bokelberg, E., Chen, Y., Kash, I. A., Parkes, D. C., Rao, M., and Sridharan, M. (2010).

Software economies. In *Proc. FSE/DSP workshop on Future of software engineering research / ACM SIGSOFT Conf.*, pages 7–12.

Barbosa Lavarda, R. A., Canet-Giner, M. T., and Peris-Bonet, F. J. (2010). How middle managers contribute to strategy formation process: connection of strategy processes and strategy practices. *Revista de Administración de Empresas*, 50(4):358–370.

Beecham, S., Baddoo, N., Hall, T., Robinson, H., and Sharp, H. (2008). Motivation in software engineering: A systematic literature review. *Information and Software Technology*, 50(9-10):860–878.

Belhadi, A., Kamble, S., Jabbour, C. J. C., Gunasekaran, A., Ndubisi, N. O., and Venkatesh, M. (2021). Manufacturing and service supply chain resilience to the covid-19 outbreak: Lessons learned from the automobile and airline industries. *Technological Forecasting & Social Change*, 163:1–19.

Betz, U. A. K., Betz, F., Kim, R., Monks, B., and Phillips, F. (2019). Surveying the future of science, technology and business – a 35 year perspective. *Tecnological Forecasting & Social Change*, 144:137–147.

Boehm, B. (1981). *Software Engineering Economics*. Prentice Hall.

Boehm, B. (2000). Cocomo II model definition manual. Manual, Center for Software Engineering, University of Southern California.

Braude, E. J. and Bernstein, M. E. (2011). *Software Engineering: Modern Approaches*. John Wiley & Sons, NJ, USA, 2^{nd} edition.

Brem, A., Viardot, E., and Nylund, P. A. (2021). Implications of the coronavirus (covid-19) outbreak for innovation: Which technologies will improve our lives? *Technological Forecasting & Social Change*, 163:1–7.

Britto, R. (2015). *Knowledge Classification for Supporting Effort Estimation in Global Software Engineering Projects*. Licentiate dissertation, Blekinge Institute of Technology, Sweden.

Britto, R., Mendes, E., and Börstler, J. (2015). An empirical investigation on effort estimation in

agile global software development. In *IEEE 10th International Conference on Global Software Engineering*, pages 38–45.

Bruegge, B. and Dutoit, A. H. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java^{TM}*. Pearson Education, NJ, USA, 3rd edition.

Burgelman, R. A., Floyd, S. W., Laamanen, T., Mantere, S., Vaara, E., and Whittington, R. (2018). Strategy processes and practices: Dialogues and intersections [special issue article]. *Strategic Management Journal*, pages 1–28.

Busu, M. (2018). Game theory in strategic management - dynamic games: Theoretical and practical examples. *Management Dynamics in the Knowledge Economy*, 6(4):645–654.

Caine, J. and Von Rosing, M. (2018). Introducing the strategy lifecycle: using ontology and semiotics to interlink strategy design to strategy execution. In *IFIP Advances in Information and Communication Technology*, pages 136–144. Springer.

Cantala, D. (2014). *Émile Borel and the Notion of Strategy*. El Colegio de México, Centro de Estudios Económicos, México, D.F., 1st edition.

Carrión Maroto, J. (2007). *Estrategia De la visión a la acción*. Alfaomega ESIC, Madrid, España, 2nd edition.

Čelar, S., Turić, M., and Vicković, L. (2014). Method for personal capability assessment in agile teams using personal points. In *Proc. 22nd Telecommunications Forum Telfor / TELFOR Conf.*, pages 1134–1137.

Chudzicka, J. (2013). Insourcing as a new trend in global business. *Foundations of Management*, 5:7–24.

Ciolkowski, M., Guzmán, L., Trendowicz, A., and Salfner, F. (2017). Lessons learned from the prodebt research project on planning technical debt strategically. In *Lecture Notes in Computer Science, vol 10611. Product-Focused Software Process Improvement, PROFES 2017*, pages 523–534.

Clark, D. N. (1997). Strategic management tool usage: a comparative study. *Strategic Change*, 6:417–427.

Commeyne, C., Abran, A., and Djouab, R. (2016). Effort estimation with story points and cosmic function points – an industry case study [position paper]. *Software Measurement News*, 21(1):25–36.

Conoscenti, M., Besner, V., Vetro, A., and Méndez Fernández, D. (2019). Combining data analytics and developers feedback for identifying reasons of inaccurate estimations in agile software development. *Journal of Systems and Software*, 156:126–135.

Corgnet, B., Hernán-González, R., and Rassenti, S. (2015). Peer pressure and moral hazard in teams: Experimental evidence. *Review of Behavioral Economics*, 2(4):379–403.

Cristal, M., Wildt, D., and Prikladnicki, R. (2008). Usage of scrum practices within a global company. In *IEEE International Conference on Global Software Engineering*.

Cuadrado-García, J. L., Cuadrado-Gallego, J. J., Herranz-Martínez, M. A., and Rodríguez-Soria, P. (2011). Improve tracking in the software development projects. In *Proc. 21st International Workshop on Software Measurement / 6th International Conference on Software Process and Product Measurement*, pages 215–220.

Cunha, J. A. O. G., Moura, H. P., and Vasconcellos, F. J. S. (2016). Decision-making in software project management: A systematic literature review. *Procedia Computer Science*, 100:947–954.

Czarnacka-Chrobot, B. (2012). What is the cost of one ifpug method function point? – case study. In *The 11th International Conference on Software Engineering Research and Practice (SERP'12)*.

Dekkers, C. (2017). Ifpug 30 years – international year of software measurement timeline. *Metric Views: Celebrating IFPUG's* 30^{th} *Anniversary*, 11(2):4–5.

Desjardins, C. (2012). The leadership productivity model. *Journal of Applied Leadership and Management*, 1:20–38.

Diebold, P., Ostberg, J.-P., Wagner, S., and Zendler, U. (2015). What do practitioners vary in using scrum? In Lassenius, C., Dingsøyr, T., and Paasivaara, M., editors, *Agile Processes in Software Engineering and Extreme Programming XP 2015*, Lecture Notes in Business Information Processing, vol 212, pages 40–51. Springer.

Digital.ai. (2020). 14^{th} annual state of agile report.

Dragicevic, S., Celar, S., and Turic, M. (2017). Bayesian network model for task effort estimation in agile software development. *The Journal of Systems and Software*, 127:109–119.

Eisenhardt, K. M. (1989). Agency theory: An assessment and review. *Academy of Management Review*, 14:57–74.

Eloranta, V., Koskimies, K., and Mikkonen, T. (2016). Exploring scrumbut – an empirical study of scrum anti-patterns. *Information and Software Technology*, 74:194–203.

Fayezi, S., O'Loughlin, A., Zutshi, A., Sohal, A., and Das, A. (2019). What impacts do behavior-based and buffer-based management mechanisms have on enterprise agility? *Journal of Manufacturing Technology Management*, 31(1):169–192.

Feijs, L. (2001). Prisoner's dilemma in software testing. In *Proc. 7e Nederlandse Testdag*, pages 65–80.

Flood, M. M. (1952a). On game-learning theory and some decision-making experiments. Technical report, The Rand Corporation.

Flood, M. M. (1952b). Some experimental games. Research memorandum, The Rand Corporation.

Fustik, V. (2017). The advantages of agile methodologies applied in the ict development projects. *International Journal on Information Technologies & Security*, 4:51–62.

Galorath, D. D. (2006). The 10 step software estimation process for sucessful software planning, measurement and control.

Gavidia-Calderon, C., Sarro, F., Harman, M., and Barr, E. T. (2020). Game-teoretic analysis of development practices: Challenges and opportunities. *The Journal of Systems and Software*, 159:1–9.

Grapenthin, S., Poggel, S., Book, M., and Gruhn, V. (2014). Facilitating task breakdown in sprint planning meeting 2 with an interaction room: An experience report. In *Proc. 40th EUROMICRO Conference on Software Engineering and Advanced Applications / ACM SEAA Conf.*, pages 1–8.

Grimstad, S. and Jørgensen, M. (2007). Inconsistency of expert judgment-based estimates of software development effort. *The Journal of Systems and Software*, 80:1770–1777.

Gulliksen Stray, V., Moe, N. B., and Dingsøyr, T. (2011). Challenges to teamwork: A multiple case study of two agile teams. In Sillitti, A., Hazzan, O., Bache, E., and Albaladejo, X., editors, *Agile Processes in Software Engineering and Extreme Programming XP 2011*, Lecture Notes in Business Information Processing, vol 77, pages 146–161. Springer Berlin Heidelberg.

Hacaloglu, T. and Demirors, O. (2018). Challenges of using software size in agile software development: A systematic literature review. *International Workshop on Software Measurement IWSM Mensure Conference*, 2207:109–122.

Haugen, N. C. (2006). An empirical study of using planning poker for user story estimation. *IEEE Proceedings of AGILE 2006 Conference*.

Hazzan, O. and Dubinsky, Y. (2005). Social perspective of software development methods: The case of the prisoner dilemma and extreme programming. In *International Conference on Extreme Programming and Agile Processes in Software Engineering*, pages 74–81. Springer.

Hernández Sampieri, R., Fernández Collado, C., and Baptista Lucio, P. (2014). *Metodología de la Investigación*. McGraw Hill / Interamericana Editores, México, DF, 6^{ta} edition.

Herrero Martín, J. and Pinedo, J. (2005). Pensamiento estratégico, teoría de juegos y comportamiento humano. *Indivisa: Boletín de estudios e investigación*, 6:37–67.

Hoda, R., Noble, J., and Marshall, S. (2013). Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, 39:422–444.

Hoegl, M. and Gemuenden, H. G. (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization Science*, 12(4):435–449.

Holmstrom, B. (1982). Moral hazard in teams. *The Bell Journal of Economics*, 13(2):324–340.

Holtzhausen, N. and de Klerk, J. J. (2018). Servant leadership and the scrum team's effectiveness. *Leadership & Organization Development Journal*, 39:873–882.

Hossain, E., Babar, M. A., and Paik, H. (2009). Using scrum in global software development: A systematic literature review. In *Fourth IEEE International Conference on Global Software Engineering*.

Huanca, L. M. and Oré, S. B. (2016). Factores que afectan la precisión de la estimación del esfuerzo en proyectos de software usando puntos de casos de uso. *Revista Ibérica de Sistemas y Tecnologías de Información*, 21(3):18–32.

Ibraigheeth, M. and Fadzli, S. (2019). Core factor for software projects success. *International Journal of Informatics Visualization*, 3(1):69–74.

Idri, A., Hosni, M., and Abran, A. (2016). Systematic literature review of ensemble effort estimation. *The Journal of Systems and Software*, 118:151–175.

Jaspan, C. and Sadowski, C. (2019). No single metrics captures productivity. In Sadowski, C. and Zimmermann, T., editors, *Rethinking Productivity in Software Engineering*, pages 13–20. Springer.

Jørgensen, M. (2016a). A survey on the characteristics of projects with success in delivering client benefits. *Information and Software Technology*, 78:83–94.

Jørgensen, M. (2016b). Unit effects in software project effort estimation: Work-hours gives lower effort estimates than workdays. *The Journal of Systems and Software*, 117:117–274.

Jørgensen, M. (2019). Relations between project size, agile practices and successful software development. *Cristin*.

Jørgensen, M. and Boehm, B. (2009). Software development effort estimation: Formal models or expert judgment? [viewpoints]. *IEEE Software*, Marzo/Abril 2009.

Jørgensen, M. and Grimstad, S. (2011). The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment. *IEEE Transactions on Software Engineering*, 37(5):695–707.

Jørgensen, M. and Halkjelsvik, T. (2010). The effects of request formats on judgment-based effort estimation. *The Journal of Systems and Software*, 83:29–36.

Jørgensen, M. and Sjøberg, D. I. K. (2004). The impact of customer expectation on software development effort estimates. *International Journal of Project Management*, 22(4):317–325.

Karabiyik, T., Jaiswal, A., Thomas, P., and Magana, A. J. (2020). Understanding the interactions

between the scrum master and the development team: A game-theoretic approach. *Mathematics*, 8:1553.

Kautz, K. and Nielsen, B. (2006). Principals, agents and prisoners: An economical perspective on information systems development practice. *Issues in Informing Science and Information Technology*, 3:308–318.

Keil, P. (2005). Principal agent theory and its application to analyze outsourcing of software development. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–5.

Kitawa, N., Hata, H., Ihara, A., Kosigo, K., and Matsumoto, K. (2016). Code review participation: Game theoretical modeling of reviewers in gerrit datasets. In 9^{th} *International Workshop on Cooperative and Human Aspects of Software Engineering / CHASE'16*, pages 64–67.

Ko, A. J. (2019). Why we should not measure productivity. In Sadowski, C. and Zimmermann, T., editors, *Rethinking Productivity in Software Engineering*, pages 21–26. Springer.

KPMG (2017). Project management survey 2017: Driving business performance.

Lavazza, L. and Meli, R. (2014). An evaluation of simple function point as a replacement of ifpug function point. *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, pages 196–206.

Lee, D. and Kim, B. C. (2013). Motivations for open source project participation and decisions of software developers. *Computational Economics*, 41:31–57.

Lenarduzzi, V., Lunesu, I., Matta, M., and Taibi, D. (2015). Functional size measures and effort estimation in agile development: A replicated study. In Lassenius, C., Dingsøyr, T., and Paasivaara, M., editors, *Agile Processes in Software Engineering and Extreme Programming XP 2015*, Lecture Notes in Business Information Processing, vol 212, pages 105–116. Springer.

Lichtenstein, Y. (2004). Puzzles in software development contracting. *Communications of the ACM*, 47(2):61–65.

López-Martin, C. (2011). Applying a general regression neural network for predicting development effort of short-scale programs. *Neural Comput & Applic*, 20:389–401.

Lundene, K. and Mohagheghi, P. (2018). How autonomy emerges as agile cross-functional teams mature. In *19th International Conference on Agile Software Development / XP Conf.*, pages 1–5.

Lundin, R. and Söderholm, A. (1995). A theory of the temporary organization. *Scandinavian Journal of Management*, 11(4):437–455.

Mahnic, V. and Hovelja, T. (2012). On using planning poker for estimating user stories. *The Journal of Systems and Software*, 85:2086–2095.

Manifesto, A. (2001). Manifesto for agile software development.

Marçal, A. S. C., de Freitas, B. C. C., Soares, F. S. F., Furtado, M. E. S., Maciel, T. M., and Belchior, A. D. (2008). Blending scrum practices and cmmi project management process areas. *Innovations Syst Softw Eng*, 4:17–29.

McLeod, L. and MacDonell, S. G. (2011). Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys*, 43(4).

Meli, R. (2011). Simple function point: a new functional size measurement method fully compliant with ifpug 4.x". *8th Software Measurement European Forum - SMEF2011*.

Melo, C., Santana, C., and Kon, F. (2012). Developers motivation in agile teams. In *38th Euromicro Conference on Software Engineering and Advanced Applications*, pages 376–383.

Meyer, A., Murphy, G., Fritz, T., and Zimmermann, T. (2019). Developers' diverging perceptions of productivity. In Sadowski, C. and Zimmermann, T., editors, *Rethinking Productivity in Software Engineering*, pages 137–146. Springer.

Mintzberg, H. (1978). Patterns in strategy formation. *Management Science*, 24(9):934–948.

Mintzberg, H. (1987a). The strategy concept I: Five Ps for strategy. *California Management Review*, 30(1):11–24.

Mintzberg, H. (1987b). The strategy concept II: Another look at why organizations need strategies. *California Management Review*, 30(1):25–32.

Mintzberg, H. and Lampel, J. (1999). Reflecting on the strategy process. *Sloan Management Review*, 40(3):21–30.

Mintzberg, H. and Waters, J. (1985). Of strategies, deliberate and emergent. *Strategic Management Journal*, 6(3):257–272.

Moe, N. B. (2013). Key challenges of improving agile teamwork. In Baumeister, H. and Weber, B., editors, *Agile Processes in Software Engineering and Extreme Programming XP 2013*, Lecture Notes in Business Information Processing, vol 149, pages 76–90. Springer Berlin Heidelberg.

Moe, N. B., Aurum, A., and Dybå, T. (2012). Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology*, 54:853–865.

Moe, N. B., Dingsøyr, T., and Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, 52:480–491.

Moe, N. B., Dingsøyr, T., and Røyrvik, E. (2009). Putting agile teamwork to the test – an preliminary instrument for empirically assessing and improving agile software development. In Abrahamsson, P., Marchesi, M., and Maurer, F., editors, *Agile Processes in Software Engineering and Extreme Programming XP 2009*, Lecture Notes in Business Information Processing, vol 31, pages 114–123. Springer Berlin Heidelberg.

Moe, N. B., Šmite, D., Hanssen, G. K., and Barney, H. (2014). From offshore outsourcing to insourcing and partnerships: four failed outsourcing attempts. *Empirical Software Engineering*, 19:1225–1258.

Mohagheghi, P. and Jørgensen, M. (2017). What contributes to the success of it projects? an empirical study of it projects in the norwegian public sector. *Journal of Software*, 12:751–758.

Moløkken-Østvold, K. and Jørgensen, M. (2005). Expert estimation of web-development projects: Are software professionals in technical roles more optimistic than those in non-technical roles? *Empirical Software Engineering*, 10:7–29.

Munien, I. and Telukdarie, A. (2021). Covid-19 supply chain resilience modelling for the dairy industry. *Procedia Computer Science*, 180:591–599.

Mutiullah, J., Ayesha, W., Muhammed, S., Sherjeel, I., Muhammed, A., and Adnan, K. (2018). A

review of popular agile software development technologies. *Journal of Information Technology & Software Engineer*, 8(4).

Naik, N. (2016). Crowdsourcing, open-sourcing, outsourcing and insourcing software development: A comparative analysis. In *IEEE Symposium on Service-Oriented System Engineering SOSE*, pages 380–385.

Nash, J. (1951). Non-cooperative games. *Annals of Mathematics*, 54(2):286–295.

Nooraie, M. (2012). Factors influencing strategic decision-making processes. *International Journal of Academic Research in Business and Social Sciences*, 2(7):405–429.

Omar, A., Bass, J. M., and Lowit, P. (2016). Exploring the factors that influence the success of insourced government ict projects. *The Electronic Journal of Information Systems in Developing Countries*, 77:1–22.

Papulova, Z. and Gazova, A. (2016). Role of strategic analysis in strategic decision-making. *Procedia Economics and Finance*, 39:571–579.

Paramanantham, P., Nizam, I., and Eissa, A. M. K. (2019). The factors affecting it insourcing in the financial and oil & gas industry in malaysia. *International Journal of Information System and Engineering*, 7:57–85.

Phannachitta, P., Keung, J., Monden, A., and Matsumoto, K. (2017). A stability assessment of solution adaptation techniques for analogy-based software effort estimation. *Empirical Software Engineering*, 22:474–504.

Popli, R. and Chauhan, N. (2013). Research challenges of agile estimation. *International Journal of IT & Knowledge Management*, 7(1):108–111.

Popli, R. and Chauhan, N. (2014). Agile estimation using people and project related factors. In *International Conference on Computing for Sustainable Global Development*, pages 564–569.

Pospieszny, P., Czarnacka-Chrobot, B., and Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *The Journal of Systems and Software*, 137:184–196.

Pressman, R. S. (2010). *Ingeniería del software. Un enfoque práctico.* McGraw Hill, Nueva York, NY, 7^{ma} edition.

Prieto-Bustamante, F. (2020). Estimación de esfuerzo en desarrollo de software ágil: Estudio del estado actual en bogotá. *iTeckne*, 17:110–131.

Quinn, R. E., Faerman, S. R., Thompson, M. P., and McGrath, M. R. (1996). Becoming a master manager: A competency framework.

Qureshi, M. R. J. and Alsulami, N. (2014). Mitigating coordination costs in global software development using scrum. *International Journal of Information Engineering and Electronic Business*, 6(3):16–21.

Radu, L. (2019). Effort prediction in agile software development with bayesian networks. *Proceedings of the* 14^{th} *International Conference on Software Technologies*, pages 238–245.

Raghu, T., Jayaraman, B., and Rao, H. R. (2004). Toward an integration of agent- and activity-centric approaches in organizational process modeling: Incorporating incentive mechanisms. *Information Systems Research*, 15(4):316–335.

Raith, F., Richter, I., and Klinker, G. (2013). Identification of inaccurate effort estimates in agile software development. In 20^{th} *Asia-Pacific Software Engineering Conference (APSEC)*, pages 67–72.

Ramos, A. B. and Vilela Junior, D. C. (2017). A influência do papel do scrum master no desenvolvimento de projectos scrum. *Revista de Gestão e Projetos-GeP*, 8:80–99.

Sabbagh, R. (2013). Scrum gestão ágil para projetos de sucesso.

Satapathy, S. M. and Rath, S. K. (2017). Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations Syst Softw Eng*, 13:191–200.

Schweighofer, T., Kline, A., Pavlic, L., and Hericko, M. (2016). How is effort estimated in agile software development projects? In *Proceedings of the SQAMIA 2016:* 5^{th} *Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, pages 73–80.

Sehra, S. K., Brar, Y. S., Kaur, N., and Sehra, S. S. (2017). Research patterns and trends in software effort estimation. *Information and Software Technology*, 91:1–21.

Sharma, A. and Karambir. (2016). Empirical validation of random forest for agile software effort estimation based on story points. *International Journal of Engigeering Sciences & Research Technology*, 5(7):1437–1446.

Shastri, Y., Hoda, R., and Amor, R. (2021). The role of the project manager in agile software development projects. *Journal of Systems and Software*, 173:110871.

Šmite, D., Wohlin, C., Galviņa, Z., and Prikladnicki, R. (2014). An empirically based terminology and taxonomy for global software engineering. *Empirical Software Engineering*, 19:105–153.

Sommerville, I. (2011). *Ingeniería de software*. Pearson, 9^{na} edition.

Spiegler, S. V., Graziotin, D., Heinecke, C., and Wagner, S. (2020). A quantitative exploration of the 9-factor theory: Distribution of leadership roles between scrum master and agile team. In *International Conference on Agile Software Development*, pages 162–177. Springer.

Srivastava, P. and Jain, S. (2017). A leadership framework for distributed self-organized scrum teams. *Team Performance Management*, 23:293–314.

Streb, J. M. (2015). *Nash, el último fundador de la teoría de juegos, y la evolución del concepto de equilibrio desde Cournot*. Universidad del CEMA, Buenos Aires, Argentina, 1^{st} edition.

Sutherland, J., Harrison, N., and Riddle, J. (2014). Teams that finish early accelerate faster: A pattern language for high performing scrum teams. In 47^{th} *Hawaii International Conference on System Sciences*, pages 4722–4728.

Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. (2007). Distributed scrum: Agile project management with outsourced development teams. In 40^{th} *Hawaii International Conference on Software Systems*.

Tanveer, B., Guzmán, L., and Engel, U. M. (2017). Effort estimation in agile software development: Case study and improvement framework [special issue paper]. *Journal of Software: Evolution and Process*, 29(e1862).

The-Standish-Group (2015). Chaos report 2015.

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., and Mejías, M. (2015). Estimating, planning and managing agile web development projects under a value-based perspective. *Information and Software Technology*, 61:124–144.

Trendowicz, A. and Jeffery, R. (2014). Basic estimation strategies. In Trendowicz, A. and Jeffery, R., editors, *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Sucess*, pages 125–152.

Treude, C. and Figueira Filho, F. (2019). How team awareness influences perceptions of developer productivity. In Sadowski, C. and Zimmermann, T., editors, *Rethinking Productivity in Software Engineering*, pages 169–178. Springer.

Tucker, A. W. (1983). The mathematics of tucker: A sampler. *The Two-Year College Mathematics Journal*, 14(3):228–232.

Usman, M., Britto, R., Damm, L., and Börstler, J. (2018). Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology*, 99:21–40.

Usman, M., Mendes, E., and Börstler, J. (2015). Effort estimation in agile software development: a survey on the state of the practice. In *Proceedings of the* 19^{th} *International Conference on Evaluation and Assessment in Software Engineering EASE*, pages 1–10.

Vaitkevičius, S., Merkys, G., and Savanevičiene, A. (2006). Model of strategic analysis tools typology. *Engineering Economics*, 2(47):99–109.

Villegas Gómez, E., Ruiz Rodríguez, J. M., and López Gallego, F. (2016). El conflicto en el desarrollo ágil: una perspectiva desde el scrum. *Revista Gestión y Región*, 21:121–149.

Von Neumann, J. (1928). *Theory Minimax*. Princeton University Press, Princeton.

Wachnik, B. (2015). Moral hazard in it project completion: A multiple study analysis case. *Proceedings of the Federated Conference on Computer Science and Information Systems*, 5:1557–1562.

Weimar, E., Nugroho, A., Visser, J., Plaat, A., Goudbeek, M., and Schouten, A. P. (2017). The influence of teamwork quality on software team performance. preprint arXiv:1701.06146.

Wen, J., Li, S., Lin, Z., Hu, Y., and Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54:41–59.

Whittington, R. (1996). Stategy as practice. *Long Range Planning*, 29(5):731–735.

Wickramaarachchi, D. and Lai, R. (2017). Effort estimation in global software development - a systematic review. *Computer Science and Information Systems*, 14(2):393–421.

Yadav, V. (2016). A flexible management approach for globally distributed software projects. *Global Journal of Flexible Systems Management*, 17(1):29–40.

Yu, W., Zhao, G., Liu, Q., and Song, Y. (2021). Role of big data analytics capability in developing integrated hospital supply chains and operational flexibility: An organizational information processing theory perspective. *Technological Forecasting & Social Change*, 163:120417.

Zahraoui, H. and Janati Idrissi, M. A. (2015). Adjusting story points calculation in scrum effort & time estimation. In *10th International Conference on Intelligent Systems: Theories and Applications SITA*, pages 1–8.

Zare, F., Zare, H. K., and Fallahnezhad, M. S. (2016). Software effort estimation based on the optimal bayesian belief network. *Applied Soft Computing*, 49:968–980.

Ziauddin, S. K. T. and Zia, S. (2012). An effort estimation model for agile software development. *Advances in Computer Science and its Applications*, 2:314–324.