Ciencias Administrativas Teoría y Praxis
Núm. 2 Año 18, julio-diciembre 2022, pp. 56-69
DOI: https://doi.org/10.46443/catyp.v18i2.312

Desarrollo interno de proyectos de software con equipos maduros: estimación de esfuerzo y provisión de recursos en equilibrio
Insourcing software projects with mature teams: effort estimation and resource provision at equilibrium

María-Guadalupe Medina-Barrera[1], Rosa María Cantón Croda[2], Damián Emilio,Gibaja-Romero[3]

Resumen

La Estimación de Esfuerzo (EE) es crucial para la planeación de proyectos de software toda vez que contribuye al logro de objetivos. Sin embargo, la EE es un proceso complejo aún en las metodologías ágiles debido a los factores ambientales y estructurales que se encuentran presentes durante la interacción entre el equipo de desarrollo y su líder. Para hacer más sencillo el proceso de EE, las compañías prefieren el desarrollo interno con equipos maduros y un líder que provea recursos. Modelamos la interacción entre el líder y el equipo de desarrollo como un juego líder-seguidor para entender como ambos se comportan en equilibrio. Después, comparamos las estrategias en equilibrio del líder y del equipo de desarrollo al momento en que intercambian sus roles. Nuestros resultados principales proveen condiciones que garantizan la unicidad de las estrategias en equilibrio, y mediante ejemplos numéricos ilustramos el impacto de las variables exógenas sobre las estrategias en equilibrio.

Palabras clave: *Desarrollo de software ágil, estrategias óptimas, estimación de esfuerzo, Scrum.*

Abstract

Effort estimation (EE) is crucial for planning software projects since it contributes to delivery goals. Nevertheless, even in agile methodologies, EE is a complex process due to environmental and structural factors surrounding the interaction between the leader and the development team. To simplify EE, companies prefer insourcing development with a mature team and a leader that provides resources. We model the interaction between the leader and the development team as a leader-follower game to understand how they behave at equilibrium. Later, we compare leader and development team equilibrium strategies when they interchange their roles. Our main results provide conditions that guarantee the

1 Ph.D. in Strategic Planning and Technology Management; Academic Subdirector; Tecnológico Nacional de México/ Instituto Tecnológico de Apizaco; México. Software Development, Decision Analysis and Agent-based Modelling. Correo electrónico: guadalupe.mb@apizaco.tecnm.mx ORCID - 0000-0003-3074-0029
2 Ph.D. in Computer Science; Dean of Engineering; Universidad Popular Autónoma del Estado de Puebla; México. Big Data and Machine Learning. Correo electrónico: rosamaria.canton@upaep.mx ORCID - 0000-0002-5469-8964
3 Ph.D. in Economics; Academic Director; Mathematics Department, Graduate School of Engineering; Universidad Popular Autónoma del Estado de Puebla; México. Game Theory, Mathematical Economics and Mechanism Design. Correo electrónico: damianemilio.gibaja@upaep.mx ORCID 0000-0002-3536-4117

Artículo Recibido: 24 de agosto de 2022
Artículo Aceptado: 22 de septiembre de 2022

uniqueness of equilibrium strategies, and we illustrate the impact of exogenous variables on equilibrium strategies through numerical examples.

Keywords: *Agile software development, optimal strategies, effort estimation, Scrum.*

Introduction

Scrum is the most widely used framework for agile software development (Digital.ai, 2020; Mutiullah et al., 2018; Fustik, 2017; Usman, Mendes & Börstler, 2015). In such a context, effort estimation is fundamental in *Scrum* since it is necessary for planning a *sprint,* which is a development cycle (Azanha, Argoud, Camargo Junior & Antoniolli, 2017). However, such a process remains challenging because there is a mutual dependence between the *scrum master* and the development team. On the one hand, the *scrum master* aims to produce the highest business value in each *sprint.* On the other hand, the team wants to maximize its profits by exerting some effort. Hence, Scrum is a cycle development that casts similarities with principal-agent problems, where uncertainty is attributed to a lack of communication between the leader and the team (Eisenhardt, 1989). Consequently, companies often prioritize insourcing development with mature teams because agents know each others' abilities and expertise under such a structure (Paramanantham, Nizam & Eissa, 2019; Omar, Bass & Lowit, 2016). Also, companies have control over their products and services (Chudzicka, 2013), which is necessary for businesses based on technology and innovation (Naik, 2016).

Despite the advantages of insourcing development with mature teams, EE is not an easy task because it is a complex process where agents face structural deficiencies and pursue different objectives (Popli & Chauhan, 2014). In this paper, we perform numerical simulations concerning the behavior of the scrum master and the development team at equilibrium. The simulated strategies are based on Medina-Barrera et al. (2022), which analyzes a leader-follower interaction between the scrum master (who provides resources during the first stage) and the development team (that exerts effort in the second stage). Later, they analyze two alternative scenarios where i) agents exchange their roles and ii) an additional meeting is considered. Our main contribution relies on showing the impact of parameters' variations on equilibrium strategies.

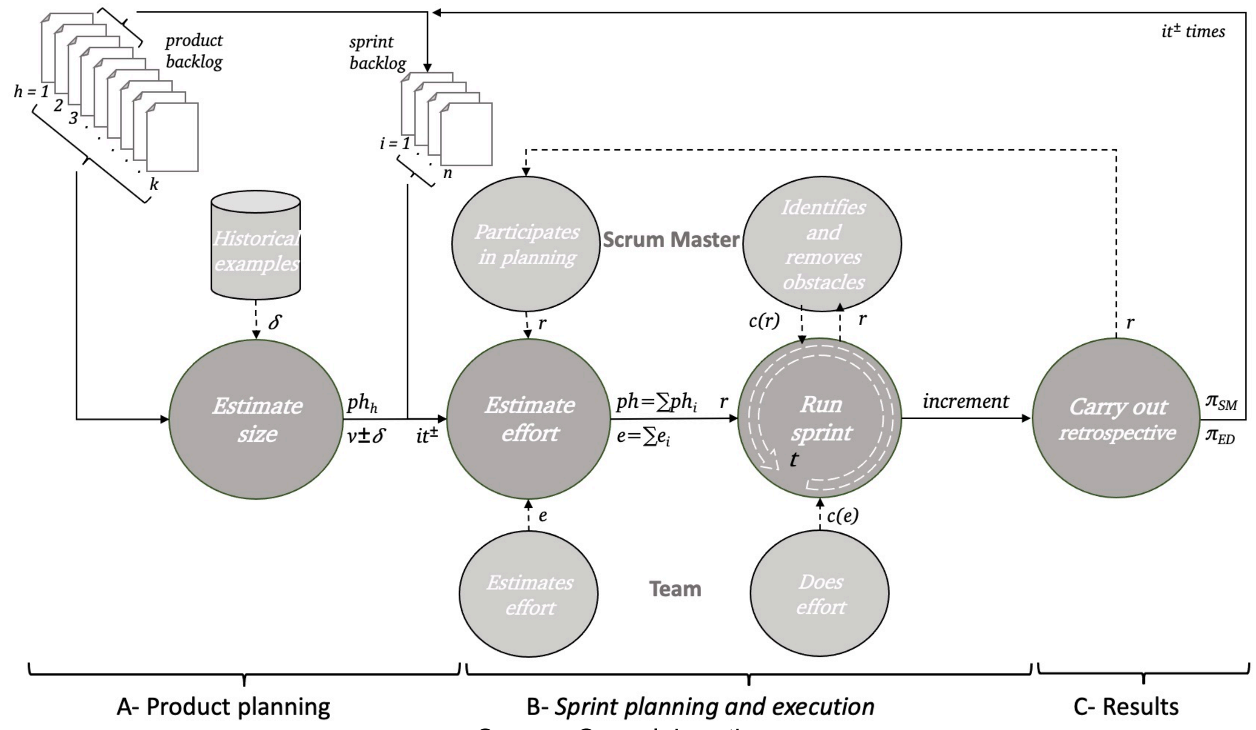
In recent years, the importance of software development analysis has increased since digital solutions diminish costs and increase efficiency. So, digital solutions are increasingly required in social and economic activities. However, development teams struggle to cope with delivery given the complexity of software projects and their increasing demands, which saturates development teams (Brem, Viardot & Nylund, 2021). So, effort estimation and resource provision are crucial for planning software projects and achieving successful results (Mohagheghi & Jørgensen, 2017; Arias et al., 2012). We observe that effort and resources at equilibrium increases as the players are more skilfull but the interaction structure reduces the resources at equilibrium when the scrum master is the leader; We also observe decreasing marginal returns finding a point where expending more effort becomes inefficient.

This paper is organized into fifth sections, as follows. The second section explains how a software project development is planned under the *Scrum* context. The third section presents the game-theoretic model of Medina-Barrera, et al. (2022) for effort estimation and resource provision in *Scrum* projects. Also, we describe the variations of such a model. The fourth section shows some numerical examples, and we derive strategies for managing software projects. Finally, the conclusions are exposed in the last section.

Product planning

Scrum develops software projects by carrying them out incrementally; in other words, the customer receives partial deliveries following planned scheduling. From the product owner, the *scrum master* gets the product backlog, a prioritized list with *k* user stories[1] describing customer requirements that the team must develop (see block A in Figure 1). So, the scrum master splits the project into parts and establishes the number of partial deliveries and their features, such as how long they will take and the deliveries' objective, which are the project's parts of being built in such a delivery.

Figure 1
Events around the EE game under Scrum context



Source. Own elaboration.

It is worth recalling that user stories' priority[2] is agreed with the customer based on its business value. Thus, the development team should address the highest priority user stories in the *product backlog* first. Then, the team estimates the size of each user story through *story points*[3] that compose scrum cycles, also known as *time-boxed* (Torrecilla-Salinas et al., 2015). Afterward, the team communicates its **iteration**

[1] In every user story, a software characteristic to be developed is described in the client's language briefly and its details will be discovered during the *sprint* (Mahnič & Hovelja, 2012; Torrecilla-Salinas et al., 2015).
[2] The user story's priority is calculated as the product of the urgency of its development and its business value, that is, the income that can be received as soon as it is available (Zahraoui & Janati Idrissi, 2015).
[3] The *story points* are a unit of measurement representing the relative size of a user story compared to the rest of the stories in the product backlog (Torrecilla-Salinas et al., 2015).

velocity *v* to the *scrum master*; *v* represents the number
of stories the team can develop in a time *t*.

The *scrum master* can fit the velocity *v* if he has historical data from similar projects developed by the team. Besides, it is necessary to set a **tolerance range** δ around *v*. The *scrum master* establishes δ with the support of the development team through a preliminary analysis of the risks involved concerning the project's development. Hence, we have that $it^{-} = k/v + \delta$ is the **minimum number of iterations or partial deliveries** of the total product, whereas $it^{+} = k/v - \delta$ is the **maximum number of iterations**. The previous process represents the initial planning for the software product's agile development.

Later, when the first sprint starts, the strategic interaction between the scrum master and the development team emerges since each of them pursues their maximum benefit. Let us explain this point. On the one hand, the *scrum master* aims to produce the highest business value during the time interval $[it^{-}, it^{+}]$; for example, the scrum master may accelerate the sprint to attend to other projects. On the other hand, exerting effort generates costs for the development team (like transportation); consequently, the development team exerts the effort that maximizes its profits.

It is worth mentioning that *v* indirectly summarizes the team's abilities because the velocity points out the team's productivity at each iteration. So, *v* should be updated at the end of each iteration, while the job must be re-estimated for the next delivery. In such a way, each iteration represents a new conflict since *v* reflects the team's current development capacity.

The EE game model

Given the previous discussion, this paper analyzes a single sprint with a fixed iteration velocity *v*. The set of players is $J = \{SM, DT\}$, where *SM* is the scrum master, and *DT is* the development team. We consider an insourcing development project in the hands of a mature team, which implies complete information. Also, we assume that *DT* makes decisions as a single player since scrum teams used to be self-organized. In other words, *DT* shares goals and makes decisions collectively (Srivastava & Jain, 2017).

The set of *DT*'s actions is $A_{DT} = \{e \in \mathbb{R} | 0 \leq e \leq t\}$, where e is the estimated effort to perform the necessary tasks for building the *story points* of the iteration in progress. Regarding *SM*'s actions, the *SM*'s mission is to support *DT* by removing obstacles during the iteration; hence, SM is a resource provider (Villegas Gómez et al., 2016; Srivastava & Jain, 2017). Consequently, the *SM*'s actions are the resources r that he provides to *DT*, that is $A_{SM} = \{r \in \mathbb{R} | 0 \leq r\}$. An actions profile is a pair $(e, r) \in A_{DT} \times A_{SM}$ that summarizes all relations in project development.

Players are characterized by a behavior type that summarizes interpersonal skills and impacts their decision-making (Ramos & Vilela Junior, 2017). The *DT*'s type h varies according to its members' knowledge, skills, and experience (Čelar et al., 2014). Concerning the scrum master, her type f represents *SM* flexibility in resource provision as a team leader (Sabbagh, 2013; Quinn et al., 1996). We assume that both types take values between 0 and 1. If $h = 0$, the *DT* does not have the necessary skills to cope with the project's objectives, while $h = 1$ states the opposite. Also, a type $f = 0$ means that the *scrum master* is not interested in providing resources to the *DT*, while $h = 1$ describes the opposite. Since we assume an insourcing development with a mature team, the types' vector (f, h) is of common knowledge.

Players' benefits are mutually dependent since exerting effort requires resources, while providing resources needs coping with stories. So, *SM*'s benefits depend on the *DT*'s effort, and the *DT*'s benefits depend on the *SM*'s support. On one side, we consider that *SM* provides *DT* with basic infrastructure K and additional resources r. On the other side, the *DT* exerts a fixed effort G related to transportation and administrative activities. Thus, the estimated effort e is additional for developing the iteration's activities.

We define players' benefits as the difference between revenues and costs. We denote the benefits of the development teams and the scrum master as $\pi_{DT} = B_{DT} - C_{DT}$ and $\pi_{SM} = B_{SM} - C_{SM}$, respectively.

To describe the previous functions, we first consider that the DT gets a monetary income $I > 0$. Moreover, the sprint's outputs result from mixing resources and effort. So, we assume that both agents have Cobb-Douglas functions that depend on the inputs basket $(r, e) \in A_{SM} \times A_{DT}$. To simplify the model, we assume that the selling price of agents' products is standardized to one. So, the revenue of each agent is

$$B_{DT}(r, e) = I + (K + r)(G + e), \qquad and \qquad B_{SM}(r, e) = (K + r)\sqrt{G + e}.$$

Players in J face costs by exerting effort and providing resources. As is common in the agile software literature, we use quadratic functions to represent players' costs (Lee & Kim, 2013). Quadratic costs are appealing for this analysis since they illustrate increasing marginal costs, which means that costs increase as the project needs an additional unit of effort or resources. Concerning the players' types, we assume that costs diminish as (h, f) improves since behavior types represent skills and disposition to perform activities in a better way. The project's environment also impacts the cost function since it serves as an adjusting parameter; that is to say, if stability τ prevails in the working place, it is easy to develop all the activities that the sprint needs. In other words, costs diminish as τ increases (Ziauddin & Zia, 2012). Finally, *DT*'s effort costs are weighted by the tasks' complexity γ involved in the current sprint backlog (Ziauddin & Zia, 2012). The previous considerations are summarized in the following costs functions:

$$c_{DT}(e) = \gamma \frac{(G + e)^2}{h} \qquad and \qquad c_{SM}(r) = \frac{(K + r)^2}{f\tau}.$$

Players interaction

The impact of *SM's* resources on DT's effort varies according to how they are deployed during the *sprint*. Hence, it is crucial to establish user stories' complexity in the *sprint backlog* and all the impediments and conflicts hindering their construction. Therefore, the *Scrum* framework outlines the following events around the *sprint*:

1. The planning meeting,
2. The daily meeting,
3. and the retrospective meeting.

During daily meetings, members of the development team answers questions like

- What have you done since yesterday?
- What are you going to do today? and
- Do you have any impediment that is not allowing you to advance?

Besides, the *SM* may monitor *DT*'s happiness during the retrospective meeting by asking questions like

- How happy are you at your job role?

The *SM's* main goal is to identify whether *DT* is facing issues or conflicts and their impact on completing the *sprint backlog*. If such impediments cannot be overcome in the current iteration, *SM* can decide to abort it and to re-plan it. In such a case, both players would receive null payoffs ($\pi_{SM}, \pi_{ED} = 0$).

These procedures are recommended patterns for developing *Scrum* projects whose responsibility lies with *SM* (Sutherland, Harrison & Riddle, 2014).

The previous meeting classification also sets up different scenarios for the interaction between the *SM* and *DT*. We formalize such scenarios as sequential games since *DT* and *SM* do not simultaneously unfold their activities. Given a fixed *sprint backlog* $\Pi = (s_1, s_2, \dots s_n)$, the common base list of activities, the interaction between *SM* and *DT* may unfold in one of the following three scenarios

Scenario 1

The scrum master and the development team take the role of leader and follower, respectively. Thus, we have the following two-stage game:

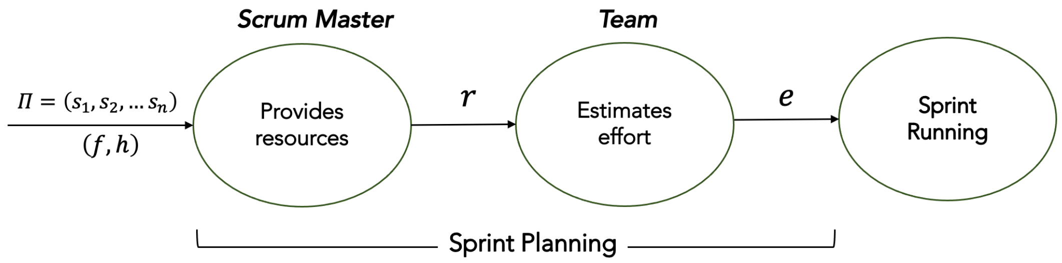
Stage 1. The *SM* chooses the additional resources r that she provides to the *DT*.

Stage 2. The *DT* observes resources that *SM* provides in the previous stage. Later, *DT* establishes the effort e to exert during this stage.

The number of resources *SM* provides to *DT* is $K + r$, i.e., basic infrastructure plus additional resources to develop the *sprint backlog*. Also, it is worth noticing that *SM* only participates in the *sprint* planning meeting. So, she ceases her interaction with *DT* (see figure 2).

Figure 2

Scenario 1: Sprint planning



Source. Own elaboration.

Scenario 2

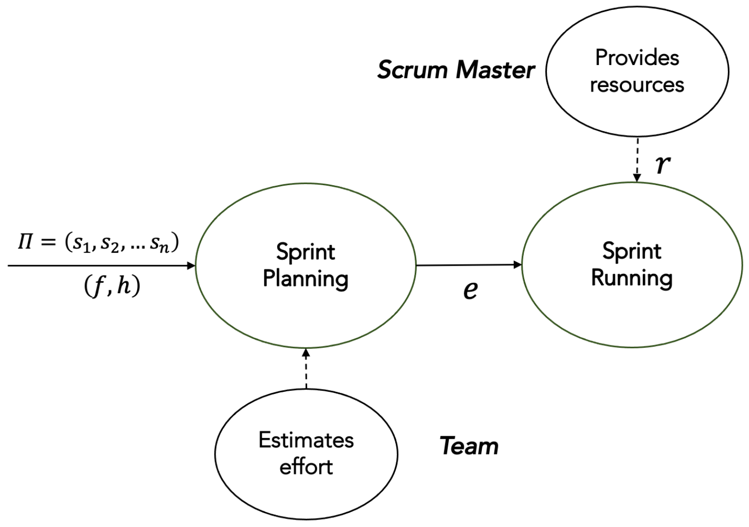
In this scenario, the players exchange their roles. Now, the development team is the leader, while the scrum master is the follower. So, the game proceeds as follows:

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

Stage 2. The *SM* observes the effort that the *DT* exerts in the previous stage and chooses the additional resources r that she provides for the sprint's activities.

In this scenario, *SM* behaves as part of the team by providing resources. Such an interaction results from close communication between them in the daily meeting to identify and eliminate obstacles during the iteration (see figure 3). We can say that the *SM* actively collaborates with *DT* to accomplish the *sprint*'s goals. Then, besides the basic infrastructure *K*, *SM* provides additional resources *r* required by *DT* to remedy conflicts arising during the sprint. Such resources may include upgraded equipment, maintenance, training, exit permissions, and free time for recreation.

Figure 3
Scenario 2: Daily meeting



Source. Own elaboration.

Scenario 3

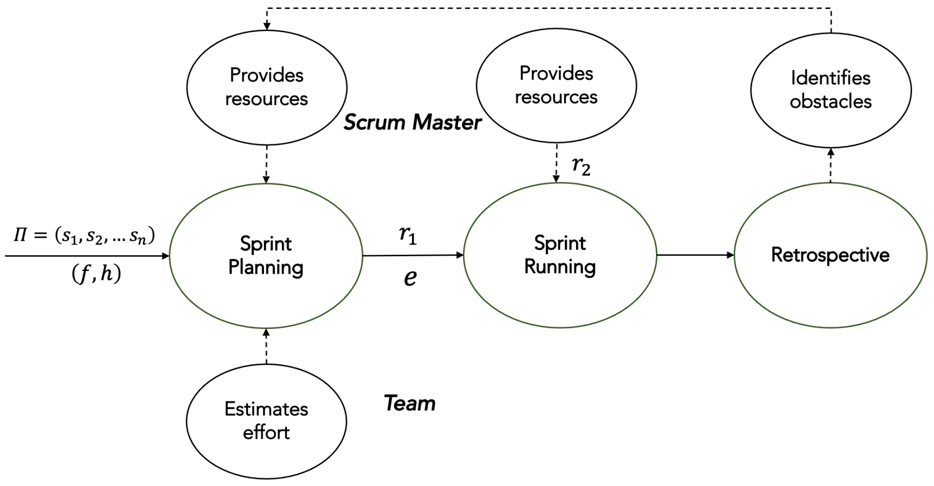
In the last scenario we consider, the *scrum master* intervenes in two stages. So, there is a planning meeting where *SM* provides initial resources and a retrospective meeting after the teams exert effort where the *SM* offers additional resources. Formally, the previous interaction is the next sequential game:

Stage 1. The *SM* chooses the initial resources r_1 that she provides at the beginning of the sprint.
Stage 2. The team observes resources r_1 and establishes the effort e to exert during the iteration.
Stage 3. After the *DT* exerts effort, the *SM* observes the sprint's state. So, *SM* provides additional resources r_2.

Then, the third scenario describes a scrum interaction where the *SM* actively participates before, during, and after the *sprint*. In other words, during the sprint planning meeting, *SM* identifies and eliminates obstacles that *DT* may face. Finally, in the retrospective meeting, *SM* identifies the main impediments to completing the *sprint*. Thus, the last meeting requires collaboration with the *DT* since the *SM* observes the team effort. Suppose the SM and the DT identify complex tasks. In that case, the discoveries are inserted as the highest priority *user story* on the subsequent *sprint* backlog (see figure 4), which is a designed pattern to scale results (Sutherland, Harrison & Riddle, 2014). In summary, the *DT* receives the basic infrastructure *K* and initial resources r_1; if they are not enough, the *SM* provides additional resources r_2.

Figure 4

Scenario 3: Retrospective meeting



Source. Own elaboration.

Finding the optimal strategies at equilibrium

It is worth emphasizing that *SM* and *DT* follow rational and strategic behavior during the sprint since both choose strategies that maximize their benefits. Still, each other decisions also impact their gains. The solution concept we study is the Subgame Perfect Nash Equilibrium to avoid non-credible strategies from both agents while coping with the sprint's goals. For *DT*, we use e^* to denote an equilibrium strategy that maximizes π_{DT}. Similarly, we say that r^* is the equilibrium strategy of the *SM*, which maximizes her benefit function π_{SM}. Formally, a strategies profile (e^*, r^*) is a subgame perfect Nash equilibrium if each strategy eliminates incentives to change strategies at each sub-game.

Medina-Barrera et al. (2022) get the equilibrium strategies through a backward induction under which each sequence of events is resolved from the last to the first stages. In other words, such a process first computes the equilibrium strategies at the final stage, and such a strategy is used to solve the previous stage. Since a single player makes decisions at each stage, equilibrium strategies solve a maximization problem, i.e., we apply the first- and second-order conditions.

Numerical examples

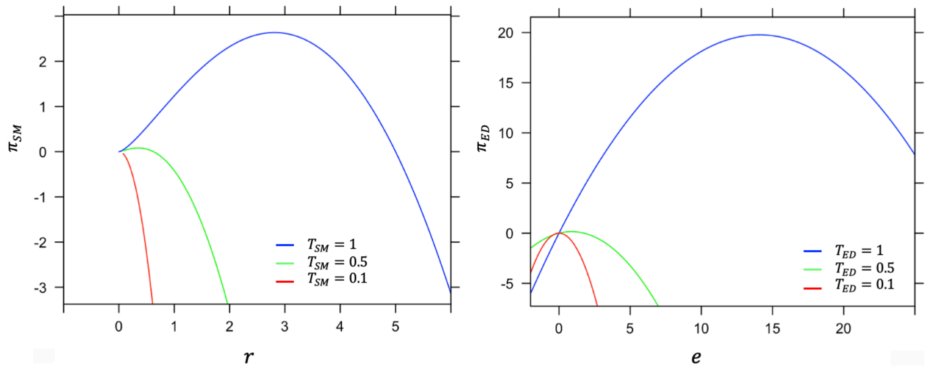
In this section, we perform some numerical examples to show how the equilibrium strategies (r^* and e^*) as well as the players' benefits at equilibrium (π_{SM} and π_{DT}) change as the exogenous parameters vary. Firstly, we study the impact of the players' type (h and f). Then, we investigate the influence of the sprint backlog complexity and the environmental stability (γ and τ).

The impact of *DT*'s experience and *SM*'s flexibility

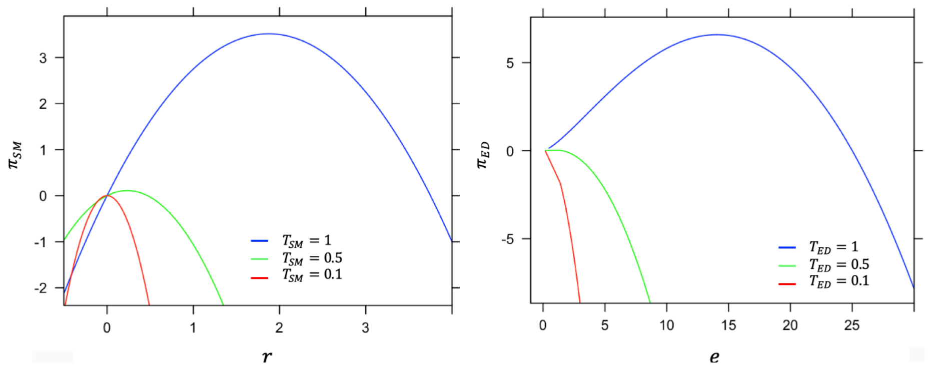
In this example, we analyze the relationship between the equilibrium strategies (r^* and e^*), benefits at equilibrium (π_{SM} and π_{DT}) concerning players' types h and f. We compare the impact of such parameters on the equilibria of each scenario. We fix the value of other exogenous. Specifically, we consider the stable environment is τ*=1*, low sprint backlog complexity γ*=0.1*, null monetary income *I=0*, null fixed expense *G=0*, and null basic infrastructure *K=0*. Figure 5 illustrates the impact of types on equilibrium strategies, and Table 1 presents the numerical results.

Figure 5.

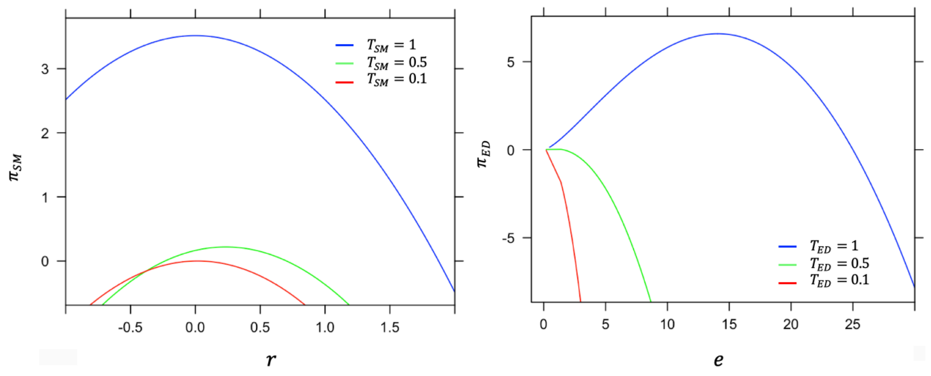
Changes in r^*, e^*, π_{SM}, π_{DT} as f and h change



a) Scenario 1



b) Scenario 2



c) Scenario 3

Source. Own elaboration.

Table 1

Players' best strategy and payoffs as f and h change by each scenario

Scenario	*r**	π_{SM}	*e**	π_{DT}
DT's low experience h=0.1 and SM's inflexible profile f=0.1				
1	0.0028	2.6367e-05	0.0014	1.9775e-06
2	0.0018	3.5156e-05	0.0014	6.5917e-07
3	0.0018	6.6796e-05	0.0014	6.5917e-07
DT's average experience h=0.5 and SM's medium profile f=0.5				
1	0.3515	0.0823	0.8789	0.1544
2	0.2343	0.1098	0.8789	0.0514
3	0.2343	0.1647	0.8789	0.0514
DT's high experience h=1 and SM's flexible profile f=1				
1	2.8125	2.6367	14.0625	19.7753
2	1.875	3.5156	14.0625	6.5917
3	1.875	3.5156	14.0625	6.5917

Source. Own elaboration.

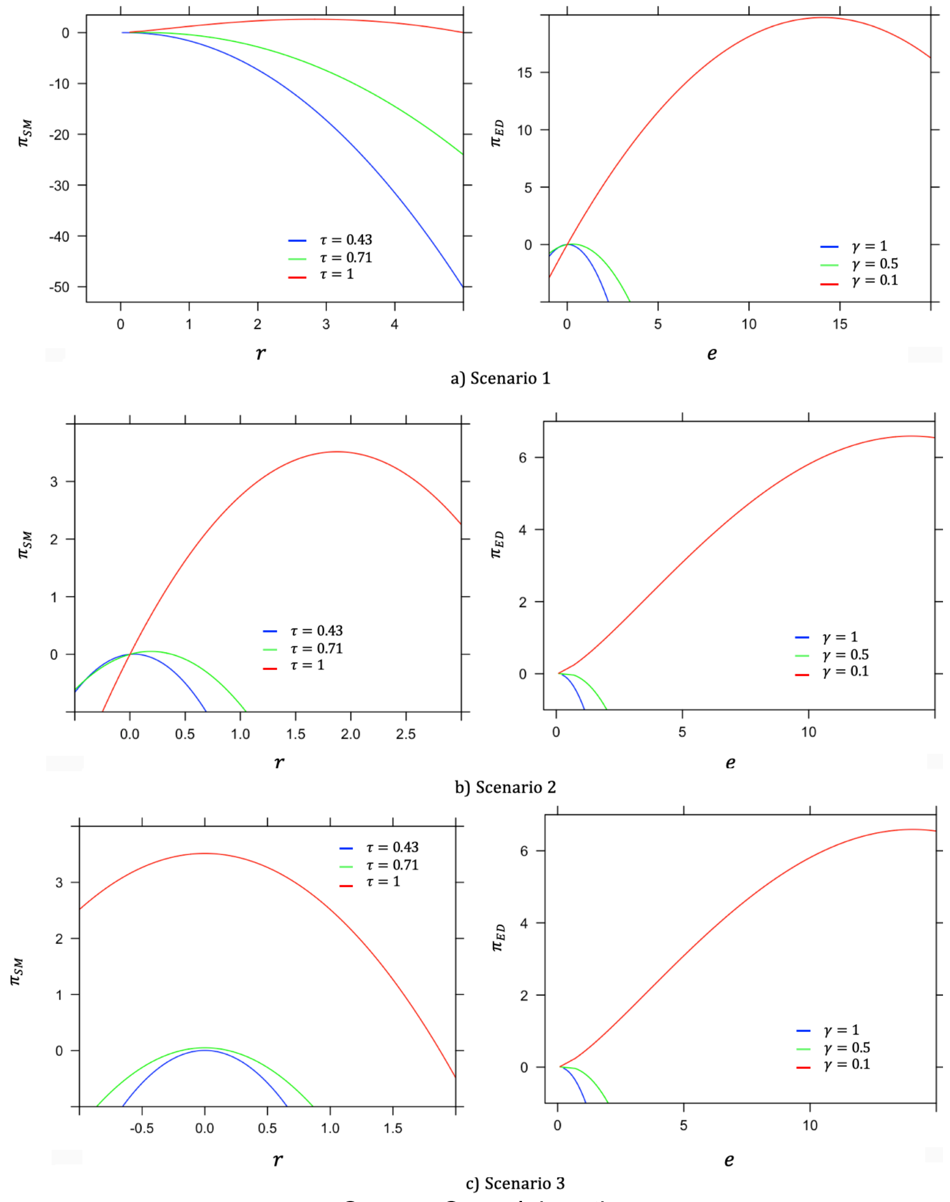
We note that players' types have a different impact on players' strategies and payoffs. SM's strategy and payoff increase as her profile becomes more flexible. The effort of DT at equilibrium increases as the DT is more skillful; as a consequence, its payoff also increases. Interestingly, the interaction structure reduces the resources that SM provides at equilibrium when SM is the leader. On the contrary, players' payoff increases while they occupy the follower position. Thus, SM can take advantage of the interaction structure where he participates as a leader and follower by deciding how much support provide to DT in each intervention. Such a scenario increases the SM's benefits, avoiding the waste of resources. These results show the advantages of holding retrospective and daily meetings to boost the sprint results.

Equilibrium strategies under variations on the Spring backlog complexity and environmental stability

Now, we analyze equilibrium when the sprint backlog complexity and the environmental change. So, we illustrate the impact of changes γ and τ in the equilibrium strategies (r^*, e^*) and the corresponding payoffs (π_{SM}, π_{DT}). We set other exogenous parameters by considering the following values: *DT* has high experience (*h=1)*, the *SM* is flexible (*f=1)*, null monetary income (*I=0)*, null fixed expense (*G=0)*, and null basic infrastructure (*K=0)*. Figure 6 shows the impact of backlog complexity and environmental stability on equilibrium strategies and benefits. Table 2 presents the corresponding numerical results.

Figure 6

Changes in r^{*}, e^{*}, π_{SM}, π_{DT} as γ and τ *change*



Source. Own elaboration.

Table 2
Players' best strategy and payoffs as γ and τ *change by each scenario*

Scenario	***r****	π_{SM}	***e****	π_{DT}
Low sprint backlog complexity γ=0.1 and stable environment τ=1				
1	2.8125	2.6367	14.0625	19.7753
2	1.875	3.5156	14.0625	6.5917
3	1.875	3.5156	14.0625	6.5917
Average sprint backlog complexity γ=0.5 and volatile environment τ=0.71				
1	0.2835	0.0377	0.2835	0.0402
2	0.1890	0.0503	0.2835	0.0134
3	0.1890	0.0503	0.2835	0.0134
High sprint backlog complexity γ=1 and highly unstable environment τ=0.43				
1	0.0520	0.0020	0.0260	0.0006
2	0.0346	0.0027	0.0260	0.0002
3	0.0346	0.0027	0.0260	0.0002

Source. Own elaboration.

In Figure 6, we generally observe non-linear behaviors, meaning that marginal effects depend on the relationship between the interaction's parameters. In other words, there is no negative or positive relationship since exogenous factors are not independent between them. Interestingly, we observe an inverse U that illustrates decreasing marginal returns in most cases. There is a point where the sprint reaches its maximum results, at which expending more effort becomes inefficient. We can say that too much stability and simplicity can discourage people from striving. In contrast, extremely complex projects developed in an excessively turbulent environment can also put people off.

Results and discussion

In this work, we model the effort estimation process in a Scrum context as a leader-follower game where players have complete information about each other. Given the importance of effort exertion to complete software projects under insourcing development, we study the interaction between a scrum master and a mature development team when they know the attributes and capabilities of each other. In such a game, the scrum master provides resources while the development team exerts effort. Later, we compare leader and development team equilibrium strategies when they interchange their roles. Our results provide conditions that guarantee the uniqueness of equilibrium strategies. We also find that equilibrium effort is the same regardless of the game structure we consider. At the same time, the scrum master has the opportunity to diminish the number of resources that she provides when she behaves as a follower. Interestingly, being a follower provides a larger payoff than being a leader for both the scrum master and the development team.

Artículo Recibido: 24 de agosto de 2022
Artículo Aceptado: 22 de septiembre de 2022

We think this could provide useful insights for agile software development environments since servant leadership may reduce the waste of resources, as the third scenario suggests.

Conclusions

This work presents a game-theoretical model to find the optimal leader-team strategies in the EE process in a Scrum context. Since EE is crucial to accomplish software projects in insourcing environments, we consider mature teams; the parties involved know each other's abilities. By analyzing three different scenarios, we find that players' strategies at equilibrium increase when the environment is stable, the team experience, and the leader flexibility are high. Meanwhile, players' strategies decrease as the sprint backlog becomes more complex. Therefore, team velocity accelerates when the sprint backlog is simple, the environment is highly stable, and the leader and the team are skillful. Here, the leader has a strategic role in achieving the sprint goals by supporting the team through the scrum meetings.

References

Arias, G., Vilches, D., Banchoff, C., Harari, I., Harari, V., & Iuliano, P. (2012). The 7 key factors to get successful results in the IT Development projects. *Procedia Technology, 5*, 199-207. doi: 10.1016/j.protcy.2012.09.022

Azanha, A., Argoud, A., Camargo Junior, J., & 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. doi: 10.1108/IJMPB-06-2016-0054

Brem, A., Viardot, E., & 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. doi: 10.1016/j.techfore.2020.120451

Čelar, S., Turić, M., & Vicković, L. (2014). Method for personal capability assessment in agile teams using personal points. *22nd Telecommunications Forum Telfor (TELFOR)*, Belgrade, Serbia, 1134-1137. doi: 10.1109/TELFOR.2014.7034607

Chudzicka, J. (2013). Insourcing as a new trend in global business. *Foundations of Management, 5*(2), 7-24. doi: 10.2478/fman-2014-0009

Digital.ai (2020). 14th annual state of agile report. *Digital.ai Software, Inc*. Retrieved from https://www.stateofagile.com

Eisenhardt, K.M. (1989). Agency Theory: An Assessment and Review. Academy of Management Review, 14(1), 57-74.

Fustik, V. (2017). The advantages of agile methodologies applied in the ICT development projects. *International Journal on Information Technologies & Security, 4*(9), 51-62.

Lee, D., & Kim, B.C. (2013). Motivations for open source project participation and decisions of software developers. *Computational Economics, 41(1)*, 31-57. doi: 10.1007/s10614-011-9311-x

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *The Journal of Systems and Software, 85*, 2086-2095. doi: 10.1016/j.jss.2012.04.005

Medina-Barrera, M.G., Cantón-Croda, R.M., & Gibaja-Romero, D.E. (2022). Effort Estimation in Insourcing Agile Software Development with Mature Teams (Working Paper). Universidad Popular Autónoma del Estado de Puebla.

Mohagheghi, P., & 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*(9), 751-758. doi: 10.17706/jsw.12.9.751-758

Mutiullah, J., Ayesha, W., Muhammed, S., Sherjeel, I., Muhammed, A., et al. (2018). A Review of Popular Agile Software Development Technologies. *Journal of Information Technology & Software Engineer, 8*(4). doi: 10.4172/2165-7866.1000245

Naik, N. (2016). Crowdsourcing, Open-Sourcing, Outsourcing and Insourcing Software Development: A Comparative Analysis. *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, March 29 – April 2, Oxford, UK. doi: 10.1109/SOSE.2016.68

Omar, A., Bass, J.M., & 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*(5), 1-22. doi: 10.1002/j.1681-4835.2016.tb00564.x

Paramanantham, P., Nizam, I., 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*(1), 57-85. doi: 10.24924/ijise/2019.04/v7.iss1/57.85

Quinn, R.E., Faerman, S.R., Thompson, M.P., & McGrath, M.R. (1996). *Becoming a master manager: A competency framework.* New York: John Wiley & Sons.

Ramos, A.B., & Vilela Junior, D.C. (2017). A Influência do Papel do Scrum Master no Desenvolvimento de Projectos Scrum. *Revista de Gestão e Projetos-GeP, 8*(3), 80-99. doi: 10.5585/gep.v8i3.556

Sabbagh, R. (2013). Scrum Master. En R. Sabbagh, *Scrum Gestão Ágil para Projetos de Sucesso* (pp. 91-106). São Paulo: Casa do Código.

Schweighofer, T., Kline, A., Pavlic, L., & Hericko, M. (2016). How is Effort Estimated in Agile Software Development Projects?. En: Z. Budimac, Z. Horváth, T. Kozsik (Eds.) *Proceedings of the SQAMIA 2016: 5th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, Budapest, Hungary, *29*, (73-80).

Srivastava, P., & Jain, S. (2017). A leadership framework for distributed self-organized scrum teams. *Team Performance Management, 23*(5/6), 293-314. doi: 10.1108/TPM-06-2016-0033

Sutherland, J., Harrison, N., & Riddle, J. (2014). Teams that finish early accelerate faster: A pattern language for high performing scrum teams. *47th Hawaii International Conference on System Sciences*, Waikoloa, Hawaii, USA, 4722-4728. doi: 10.1109/HICSS.2014.580

Torrecilla-Salinas, C.J., Sedeño, J., Escalona, M.J., & Mejías, M. (2015). Estimating, planning and managing Agile Web development projects under a value-based perspective. *Information and Software Technology, 61*, 124–144. doi: 10.1016/j.infsof.2015.01.006

Usman, M., Mendes, E., & Börstler, J. (2015). Effort estimation in agile software development: a survey on the state of the practice. *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE), Nanjing, China, 12*, 1–10.

Villegas Gómez, E., Ruiz Rodríguez, J.M., & López Gallego, F. (2016). El conflicto en el desarrollo ágil: una perspectiva desde el SCRUM. En *Revista Gestión y Región, no. 21*, (pp. 121-149). Universidad Católica de Pereira, Colombia.

Zahraoui, H., & Janati Idrissi, M. A. (2015). Adjusting story points calculation in scrum effort & time estimation. *10th International Conference on Intelligent Systems: Theories and Applications (SITA)*, Rabat, Marruecos, 1-8. doi: 10.1109/SITA.2015.7358400

Ziauddin, S.K.T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in Computer Science and its Applications, 2*(1), 314-324.