2-12-2021

Exploring the Efficiency of Self-Organizing Software Teams with Game Theory

Clay Stevens
University of Nebraska - Lincoln, clay.stevens@huskers.unl.edu

Jared Soundy
University of Nebraska-Lincoln, jared.soundy@huskers.unl.edu

Hau Chan
University of Nebraska-Lincoln, hchan3@unl.edu

Exploring the Efficiency of Self-Organizing Software Teams with Game Theory

Clay Stevens*, Jared Soundy†, and Hau Chan‡
Computer Science and Engineering, University of Nebraska–Lincoln
*clay.stevens@huskers.unl.edu, †jared.soundy@huskers.unl.edu, ‡hchan3@unl.edu
*Corresponding author

Abstract—**Over the last two decades, software development has moved away from centralized, plan-based management toward agile methodologies such as Scrum. Agile methodologies are founded on a shared set of core principles, including self-organizing software development teams. Such teams are promoted as a way to increase both developer productivity and team morale, which is echoed by academic research. However, recent works on agile neglect to consider strategic behavior among developers, particularly during task assignment–one of the primary functions of a self-organizing team. This paper argues that self-organizing software teams could be readily modeled using game theory, providing insight into how agile developers may act when behaving strategically. We support our argument by presenting a general model for self-assignment of development tasks based on and extending concepts drawn from established game theory research. We further introduce the software engineering community to two metrics drawn from game theory—the price-of-stability and price-of-anarchy—which can be used to gauge the efficiencies of self-organizing teams compared to centralized management. We demonstrate how these metrics can be used in a case study evaluating the hypothesis that smaller teams self-organize more efficiently than larger teams, with conditional support for that hypothesis. Our game-theoretic framework provides new perspective for the software engineering community, opening many avenues for future research.**

I. Introduction

The publication of the *Agile Manifesto* [1] in 2001 instigated a long-lived change in the way software development teams are managed, pushing the industry toward agile methodologies such as Scrum or lean [2]. A reported 95% of respondents in the 2020 *State of Agile* report—software professionals from across the world in industries ranging from technology to retail—reported that agile development methodologies are used in their workplace. Most (75%) use them for sprint planning and agile project management, primarily Scrum [3].

One key principle of the *Agile Manifesto* is "[t]he best architectures, requirements, and designs emerge from self-organizing teams" [1]. As organizations embrace agile software development, they eschew centralized, plan-driven project management in favor of multi-functional, autonomous teams that can quickly adapt to changing requirements without top-down planning [4]. Self-organization is a topic of much research, with a variety of studies seeking to understand roles in *self-organizing teams (SOTs)* (e.g., see [5]–[9]) and task assignment in agile project management (e.g., see [10]–[12]). These studies provide insight into the practices and benefits of SOTs, but do little to explore *strategic* self-organization, where developers consider the choices of all other developers while seeking their best personal outcome. If team members are left to choose their own tasks strategically, they may sacrifice some social good for their own utility, decreasing the *efficiency* of the team compared to a team with an ideal centralized plan. We believe there is a need for further study of strategic behavior to answer the following research questions within the software engineering domain: (A) How can we (theoretically) model and understand strategic behavior in SOTs? (B) How can we evaluate and measure the efficiency of developers' strategic behavior? (C) How do developers self-organize in real-world settings, and is it consistent with the model's predictions?

This paper advocates for the use of non-cooperative game-theoretic models and concepts to address the above questions. We argue that aspects of developer behavior in SOTs can be modeled using game theory. Game-theoretic models are used in software engineering research (see, e.g., [13], [14]) but currently only apply to other aspects of the software development lifecycle (e.g., bug fixes, testing).

To address our research questions, we outline a preliminary game-theoretic model of self-assignment of development tasks in a SOT, and provide a detailed example of how our model can be used to evaluate the efficiency of self-organization compared to ideal plan-based project management (A). We introduce the software engineering community to two measures of efficiency used in game theory research but not yet applied in the software engineering domain—the *price-of-stability (PoS)* and the *price-of-anarchy (PoA)* [15], [16]—and describe their use with a concrete realization of our model (B). We leave our third question for future human subject studies to determine specific parameters for the model (C).

We use *PoS/PoA* to evaluate the efficiency of a simulated SOT in a concrete realization of our model, based on Tullock contests [17]–[20]. Our simulation provides evidence that smaller teams may self-organize more efficiently than larger teams when considering team morale, but may be less efficient when considering team productivity. We describe our findings and provide examples of future research that could grow from our model and conclude by summarizing related work.

II. Self-Organization as a Game

Agile teams are expected to *self-organize*; the members divide roles and tasks rather than having them assigned [4]. On a typical Scrum team, self assignment is performed every one or two weeks (i.e., a *sprint*). For each sprint, the team picks a set of development tasks to perform (the *sprint backlog*). The developers then independently select tasks they will work on during the sprint from the sprint backlog. As the developers are selecting tasks for themselves, this affords an opportunity for *strategic* behavior, where each selects tasks to advance their own best interest.

A. How can we model strategic behavior in SOTs?

Strategic self-assignment of tasks among developers on a SOT can be modeled and understood by representing it as a *game* [21], [22]. In a game, there is a set N of n players, where each player $i \in N$ has an action or strategy set $\mathbf{A}_i$ prescribing their possible actions and a utility function $u_i : \mathbf{A} \to \mathbb{R}$ where $\mathbf{A} = \bigtimes_{k=1}^{n} \mathbf{A}_k$. The utility of each player depends on the player's action and the actions of other players; for our model, we represent it as a combination of a *payoff* and a *cost*.

For self-assignment of tasks, there is a set of developers (i.e, players) and a set M of tasks. The action set $\mathbf{A}_i$ of developer i corresponds to the set of assignable tasks (i.e., $\mathbf{A}_i = M$). The payoff for each player represents some monetary or social gain the developer receives from completed tasks. For example, many Scrum teams track "velocity", measured in the number of "story points" or tasks completed during a sprint [23]; having a high velocity is seen as desirable, so we assume a rational developer would seek to maximize their individual velocity. Task completion can be represented by a *threshold*, t_j, for each task j and some function $g_j : \mathbf{A} \to \mathbb{R}$ indicating whether the threshold is met. Thresholds are used in public good games [24], [25], and—in our case—can represent the story points or developer hours for the task.

Our general model for self-assignment, then, is that the expected payoff u_i for each developer $i \in N = \{1, 2, ..., n\}$ under strategy profile $\mathbf{a}$ assigning effort to each task $j \in M = \{1, 2, ..., m\}$ would be computed as follows:

$$u_i(\mathbf{a}) = \sum_{j \in M} \begin{cases} f_{i,j}(\mathbf{a}) - c_{i,j}(\mathbf{a}) & g_j(\mathbf{a}) \geq t_j \\ -c_{i,j}(\mathbf{a}) & \text{otherwise} \end{cases} \tag{1}$$

- $\mathbf{a} \in \mathbf{A}$ is a strategy profile, defined as the set of strategies for each $i \in N$ such that $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$;
- $f_{i,j}(\mathbf{a}) : \mathbf{A} \to \mathbb{R}$ is the payoff to i from j under $\mathbf{a}$;
- $c_{i,j}(\mathbf{a}) : \mathbf{A} \to \mathbb{R}$ is the cost to i w.r.t. j under $\mathbf{a}$;
- $g_j(\mathbf{a}) : \mathbf{A} \to \mathbb{R}$ is the total effort for j under $\mathbf{a}$; and
- $t_j \in \mathbb{R}$ is the threshold for successful completion of j.

B. How can we evaluate and measure the efficiency of developers' strategic behavior?

Using our model, we can find *Nash equilibria (NE)*—the main solution concept in games [21], [22]—for the self-assignment game. These are possible assignments of tasks where no developer would gain any additional payoff by changing their actions while the others remain fixed. Assuming the developers are acting rationally (i.e., seeking to maximize their own individual utility), the NE represent the task assignments the SOT would choose. Using our general model defined above, a given strategy profile, $\mathbf{a}$, is a NE if and only if each player's strategy is a *best response* to those of the other players. If $\mathbf{A}_i$ is the set of all possible strategies for player i, strategy profile $\mathbf{a}$ is a NE if and only if:

$$\forall (i \in N)\ \forall (\mathbf{a}'_i \in \mathbf{A}_i)\ u_i(\mathbf{a}) \geq u_i(\mathbf{a}'_i, \mathbf{a}_{-i}). \tag{2}$$

For each NE, we can compute metrics that represent *social goods* achieved by the team as a whole. For example, the following two metrics could be computed for self-assignment of development tasks:

- *Social Welfare (SW)*: An overall measure of the payoff obtained by the developers on the team. This is computed as the sum of the payoff among all the players in a game for a given strategy profile—$SW(\mathbf{a}) = \sum_{i \in N} u_i(\mathbf{a})$—and serves as a proxy for team morale in our SOT model.
- *Threshold Completed (TH)*: The total amount of work completed during the sprint, measured as the sum of the threshold values for each completed task. We use this metric as a proxy for the productivity of the team overall.

$$TH(\mathbf{a}) = \sum_{j \in M} w_j(\mathbf{a}) \text{ where } w_j(\mathbf{a}) = \begin{cases} t_j & g_j(\mathbf{a}) \geq t_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The social goods achieved via NE are often sub-optimal when compared to the those from centralized assignment of tasks, assuming the assigner will always choose the assignment with the optimal outcome. That is, if $\mathbf{A}$ is the set of all possible strategy profiles, $\mathbf{A}^* \subseteq \mathbf{A}$ is the set of all NE, $\alpha : \mathbf{A} \mapsto \mathbb{R}$ is an arbitrary social good metric, and $\mathbf{a} = \text{argmax}_{\mathbf{a}' \in \mathbf{A}} \alpha(\mathbf{a}')$ is the optimal strategy profile, then $\forall(\mathbf{a}^* \in \mathbf{A}^*)\ \alpha(\mathbf{a}) \geq \alpha(\mathbf{a}^*)$. The difference between the NE and the optimal assignment marks the *efficiency* of strategic behavior, and can be computed using the following measures [15], [16]:

$$\textit{Price-of-stability (PoS)} = \max_{\mathbf{a}' \in \mathbf{A}} \alpha(\mathbf{a}') / \max_{\mathbf{a}^* \in \mathbf{A}^*} \alpha(\mathbf{a}^*) \tag{4}$$

$$\textit{Price-of-anarchy (PoA)} = \max_{\mathbf{a}' \in \mathbf{A}} \alpha(\mathbf{a}') / \min_{\mathbf{a}^* \in \mathbf{A}^*} \alpha(\mathbf{a}^*) \tag{5}$$

For metrics like *SW* and *TH* defined above, the closer the *PoS/PoA* is to 1.0, the more efficient the strategic behavior.

C. How do developers self-organize in real-world settings?

Our general model captures many important aspects of SOTs, but the remaining functions—$f_{i,j}$, $c_{i,j}$, and g_j—must all be chosen with great care to accurately reflect real-world developer behavior. We present a case study using normally-distributed parameters for these functions in Section III, but further research with human subjects is needed to develop an accurate model (e.g., see [20]). A future experiment could be designed in an academic setting, using websites such as TopCoder [26], or by observing behavior in industry. Subjects could be organized in a SOT, with appropriate incentives representing the payoffs and corresponding investments by the participants representing the costs.

III. Concrete Example

Self-assignment of tasks among developers can be represented as a form of *Tullock contest* or all-pay auction [17]–[20]. In a Tullock contest, the players compete for a prize by selecting the level of effort they are willing to commit to a *contest*. The winning player for a given strategy profile receives the prize, discounted by the cost of their effort. In some variants—including our model—the payoff is given to each contestant proportional to their contribution of total effort. The *expected* payoff to a given player for any given contest is a linear function of the payoff and the strategy profile. This can be extended to comprise a set of related contests, where each player's strategy defines both *how much* effort to expend and *to which contest* the effort will be allocated [19], [27], [28]. For self-assignment of tasks on a development team, the players are the developers and the contests are the development tasks.

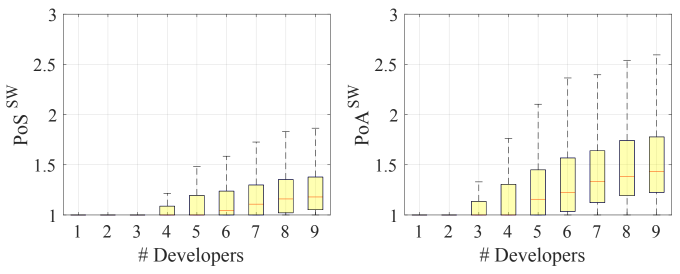
Each player selects their strategy for the game by allocating some (or no) portion of their total effort (e.g., available hours for the sprint) to the tasks. The player then receives a payoff for each *completed* task, proportional to the effort expended and discounted by the cost of their effort. The contest representing each task would provide a payoff only if the total contribution to that contest met or exceeded the threshold, but would still incur a cost to contributing players.

By using *PoS/PoA* as a measure of efficiency, we can use our model to evaluate important research questions regarding SOTs. For example, consider the "two-pizza" rule attributed to Jeff Bezos—the development team should never be so large that two pizzas cannot feed everyone. We reframe that rule as a testable hypothesis as follows:

Hypothesis 1. *Smaller development teams self-organize more efficiently than do larger teams.*

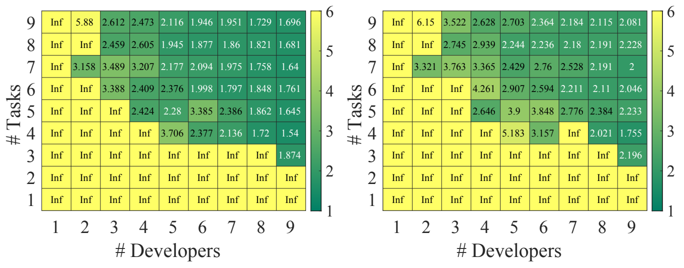
Using a concrete realization of our general model from Section II, we can compute the *PoS* and *PoA* for different team sizes and directly compare the efficiency in terms of the *SW* and *TH* metrics, which stand as proxies for team morale and productivity, respectively. To demonstrate this, we simulated our model 100 times each for every combination of $n \in [1, 9]$ and $m \in [1, 9]$, where $\mathbf{a_i}$ is a binary vector of length m with at most one 1, $f_{i,j}(\mathbf{a}) = (\mathbf{a}_{i,j} s_{i,j} v_{i,j})/g_j(\mathbf{a})$, $c_{i,j}$ was chosen randomly from a uniform distribution on $[0, 1]$, and $g_j(\mathbf{a}) = \sum_{i \in N} \mathbf{a}_{i,j}$ ($s_{i,j}$, $v_{i,j}$ are random variables representing skill and task preference, respectively). We then calculated *PoS/PoA* for each simulation. Figure 1 depicts the *PoS/PoA* as the number of developers (n) increases from 1 to 9, showing a correlated increase in the median *PoS/PoA*. This supports Hyp. 1 in terms of morale.

Considering *TH* provides a different picture. Figure 2 shows the mean *PoS/PoA* in terms of *TH* as n and m vary. By this metric, teams with fewer developers are more likely to have *unbounded PoS/PoA* (the "Inf" squares) where no tasks were completed in either the best/worst NE. This provides some indication that smaller SOTs are *not* more efficient than plan-driven teams in terms of productivity. Specifically, smaller teams (with fewer tasks) may be more likely to *not finish any tasks at all* when self-assigning than would teams where a centralized planner assigns tasks. Our model indicates that smaller teams may be efficient *when considering teams of four or more developers who are self assigning four or more tasks*. In that case, the *PoS/PoA* for *SW* is clearly lower than for larger teams (see Figure 1) and the mean *PoS/PoA* in terms of *TH* is still bounded; there is little difference in the mean *PoS/Poa* for roughly $n > 4$ and $m > 4$. This provides a demonstration of how our game-theoretical model may be used to solve software engineering questions, but a more rigorous evaluation is needed for this and other hypotheses. Furthermore, real-world human studies are also needed to validate the model against how developers actually behave.



(a) Price-of-stability (PoS) vs. n (b) Price-of-anarchy (PoA) vs. n

Fig. 1. Box plots for PoS/PoA w.r.t. social welfare (SW) vs. the number of developers (N=900 for each value on the x-axis). Median PoS/PoA rises as the number of developers increases, as does variance in PoS/PoA.



(a) Price-of-stability (PoS) (b) Price-of-anarchy (PoA)

Fig. 2. Mean (N=100) price-of-stability (PoS) and price-of-anarchy (PoA) for the *completed thresholds* (TH) vs. the number of developers and tasks. Darker (greener) show a lower PoS/PoA, indicating more efficient self-organization. Smaller teams and fewer tasks have unbounded PoS/PoA.

IV. Future Directions

The example detailed in Section II-B is only one of myriad potential findings that could come from a game-theoretical model of SOTs. The utility function u_i for the general model described in Section II can be defined in many ways, and could by used in a variety of different types of games. The *PoS/PoA* could also measure efficiency for other metrics.

Model Parameters: For example, our general model defines $f_{i,j}(\mathbf{a})$ as the expected payoff for developer i from project j in strategy profile $\mathbf{a}$. Additional parameters could be included as part of the f function to model other aspects of the real world, such as developer *skill*. McConnell [29] has proposed that developer skill can vary by up to a single order of magnitude, with the most skilled "10x" developers providing significantly more value than others. By including a variable for developer skill in our model, we could evaluate hypotheses relating to skill differences among developers or between SOTs. The payoff for a given task may also be impacted by other developers working on the same task. Brooks [30] famously described the diminishing returns of adding more developers to a single project, and Vasilescu, et al. [31] have shown that multitasking may be beneficial to productivity; both of these aspects of developer behavior could be included as part of the $f_{i,j}$ function in our model. We see many avenues for future research in developing specific functions with such parameters. Our model also assumes that the payoff for a given task may differ among the developers on the team. This could be used to model developer preferences in task selection, as was explored in a human factor study by Masood, et al. [11]. By allowing flexibility in the parameters, our general game-theoretic model can be adapted to match the findings of such human factor studies; we intend to pursue such studies in the future as a necessary validation of the model we propose here. Our model could also be extended with other parameters to represent other aspects of SOTs—negotiation among team members, overconfidence, etc.

Game Variants: Our general model could also be employed in a variety of different game formulations. In Section II-B, we describe a single realization of our model, assuming the game is a game of *complete information*. In such a game, every player knows all the possible strategies and all the *payoffs* for each such strategy for every player in the game. This assumption is often a steep hurdle for games that try to model human behavior in the real world. We also assume the developers are perfectly rational and aim to select an action that maximizes the utility, which may be incorrect. More research is needed to demonstrate our model's effectiveness for games of *incomplete* information [32] or with different bounded rationality assumptions [33], [34].

Scrum teams also tend to engage in self-organizing behavior repeatedly with the same set of players. Repeated contests behave differently than the single contests simulated in Section II-B [35], [36], and could use further study. Hoda, et al. [7] found that developers on SOTs do not all behave the same, as assumed in our simulations; the different roles assumed by the team members over time may affect their strategies when modeled as a repeated game. Our model assumes that all developers pick tasks simultaneously; in many Scrum teams, selection of individual tasks by developers is scattered throughout the sprint. These *sequential games* are also studied in game theory research [18], [37], [38], and we intend to explore their ramifications in future work. As with the different numerical parameters of the model, more real-world experiments are needed to determine which games are consistent with real-world behavior.

Social-good Metrics: Lastly, we evaluated our model in this paper against two social-good metrics—the mean *social welfare (SW)* and the *completed threshold (TH)*, standing in for team morale and productivity, respectively. The *price-of-stability/-anarchy* measures we used to evaluate efficiency could be computed for other metrics to measure the efficiency of other aspects of self-organizing teams. For example, Masood, et al. [11] found that developers often select tasks with which they are *not* particularly familiar, as a means to improve their skills. If the growth in developer skill could be quantified and measured, it could also be included as a metric in our model, providing another dimension for analysis of efficiency.

V. Related Work

Self-organization in Agile Development: Self organization in agile development has been widely reviewed [8], [9], [39]. Hoda, et al., conducted a series of surveys to classify roles that emerge on self-organizing teams [5]–[7], [10]. Our model does not consider these roles, but could be extended to account for these variations. Masood, et al., also conducted human factor studies on self-organization, including the motivations of software developers when self assigning tasks [11], [12]. As described in Section IV, motivation could be represented as parameters in our general model.

Contests and Other Games: Our model follows a stream of research on *Tullock contests* [19], [27], [28], [40] where a set of contestants determines the amount of effort to exert in either one or multiple contests in order to achieve a payoff. Congestion games [41] are also related; these present a set of resources and a set of players where each player's strategies comprise some subset of the set of resources, with the goal of minimizing wait times for use of the resource. Our model extends these games with thresholds, similar to public good games [24], [25], such that positive payoffs can only be obtained with sufficient effort.

Game Theory in SE: Game theoretic concepts have been applied to many aspects of software engineering and development, but we present the first non-cooperative model for task assignment on self-organizing teams. Lagesse [13] briefly proposed a cooperative model of task assignment, but provided no concrete evaluation. Others have used game theory to model other aspects of agile development, ranging from stand-up meetings to code reviews [42], [43]. Gavidia-Calderon, et al. [14] use game theoretic concepts to drive their mechanism design framework; we believe that price-of-stability/-anarchy presents a novel tool for mechanism design in general, and intend to explore its use in future work.

VI. Conclusion

We present a novel, general model for understanding strategic behavior of agile software developers, building off of and extending solid foundations in game theory. We introduce the *price-of-stability* and *price-of-anarchy* as well-founded metrics which the software engineering research community can employ to evaluate the efficiency of self-organizing software development teams. We describe how our general framework could be adapted to fit real-world behavior, leaving human subject studies tune the model as future work. We demonstrate the use of our model in the software engineering domain with a simulated case study. We describe a variety of new directions for future research that could spring from our model, from extensions and refinements of our general model to new human factor studies that examine developer behavior in the field. We believe our new idea provides a solid foundation for game-theoretical research in the software engineering domain.

Data Availability

The datasets generated during and/or analysed during the current study and the code used to drive the analysis are available from the corresponding author upon reasonable request.

Acknowledgment

We thank Dr. Hamid Bagheri and the anonymous reviewers for their feedback. This work was completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the Nebraska Research Initiative. This work was also supported in part by awards CCF-1755890 and CCF-1618132 from the National Science Foundation.

References

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

[2] G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical study of agile software development methodologies: A comparative analysis," *ACM SIGSOFT Software Eng. Notes*, vol. 40, no. 1, pp. 1–6, 2015.

[3] digital.ai, "14th annual state of agile report," https://stateofagile.com, 2020.

[4] A. K. Kakar, "Assessing self-organization in agile software development teams," *J. Comput. Inf. Syst.*, vol. 57, no. 3, pp. 208–217, 2017.

[5] R. Hoda, J. Noble, and S. Marshall, "Organizing *self-organizing* teams," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, J. Kramer, J. Bishop, P. T. Devanbu, and S. Uchitel, Eds. ACM, 2010, pp. 285–294.

[6] ——, "Developing a grounded theory to explain the practices of self-organizing agile teams," *Empir. Softw. Eng.*, vol. 17, no. 6, pp. 609–639, 2012.

[7] ——, "Self-organizing roles on agile software development teams," *IEEE Trans. Softw. Eng.*, vol. 39, no. 3, pp. 422–444, 2013.

[8] K. Crowston, Q. Li, K. Wei, U. Y. Eseryel, and J. Howison, "Self-organization of teams for free/libre open source software development," *Inf. Softw. Technol.*, vol. 49, no. 6, pp. 564–575, 2007.

[9] B. Acharya and R. C. Palacios, "A systematic literature review on autonomous agile teams," in *19th Int. Conf. Computational Science and Its Applications, ICCSA 2019, Saint Petersburg, Russia, Jul. 1-4, 2019, Part VII*, S. Misra, O. Gervasi, B. Murgante, E. N. Stankova, V. Korkhov, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, and E. Tarantino, Eds. IEEE, 2019, pp. 146–151.

[10] R. Hoda and L. K. Murugesan, "Multi-level agile project management challenges: A self-organizing team perspective," *J. Syst. Softw.*, vol. 117, pp. 245–257, 2016.

[11] Z. Masood, R. Hoda, and K. Blincoe, "Motivation for self-assignment: Factors agile software developers consider," in *10th IEEE/ACM Int. Workshop Cooperative and Human Aspects of Softw. Eng., CHASE@ICSE 2017, Buenos Aires, Argentina, May 23, 2017*. IEEE Comput. Soc., 2017, pp. 92–93.

[12] ——, "Exploring workflow mechanisms and task allocation strategies in agile software teams," in *Proc. 18th Int. Conf. Agile Processes in Softw. Eng. and Extreme Program., XP 2017, Cologne, Germany, May 22-26, 2017*, H. Baumeister, H. Lichter, and M. Riebisch, Eds., vol. 283, 2017, pp. 267–273.

[13] B. Lagesse, "A game-theoretical model for task assignment in project management," in *2006 IEEE Int. Conf. Manage. of Innov. and Technology, ICMIT 2006, Singapore, Jun. 21-23 2006*. IEEE Comput. Soc., 2006, pp. 678–680.

[14] C. Gavidia-Calderon, F. Sarro, M. Harman, and E. T. Barr, "Game-theoretic analysis of development practices: Challenges and opportunities," *J. Syst. Softw.*, vol. 159, 2020.

[15] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *STACS 99*, C. Meinel and S. Tison, Eds., 1999, pp. 404–413.

[16] T. Roughgarden, *Selfish Routing and the Price of Anarchy*. MIT Press, 2005.

[17] G. Tullock, "On the efficient organization of trials," *Kyklos*, vol. 28, no. 4, pp. 745–762, 1975.

[18] K. A. Konrad, *Strategy and Dynamics in Contests*. Oxford UP, 2009.

[19] H. Chan, D. C. Parkes, and K. R. Lakhani, "The price of anarchy of self-selection in tullock contests," in *Proc. 19th Int. Conf. Auton. Agents and MultiAgent Systems*, 2020, pp. 1795–1797.

[20] E. Dechenaux, D. Kovenock, and R. M. Sheremeta, "A survey of experimental research on contests, all-pay auctions and tournaments," *Experimental Econ.*, vol. 18, no. 4, pp. 609–669, 2015.

[21] J. von Neumann and O. Morgenstern, *Theory of games and economic behavior*. Princeton UP, 1947.

[22] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, ser. MIT Press Books. The MIT Press, December 1994, vol. 1, no. 0262650401.

[23] S. Downey and J. Sutherland, "Scrum metrics for hyperproductive teams: How they fly like fighter aircraft," in *46th Hawaii Int. Conf. System Sciences, HICSS 2013, Wailea, HI, USA, January 7-10, 2013*. IEEE Comput. Soc., 2013, pp. 4870–4878.

[24] Z. Komarovsky, V. Levit, T. Grinshpoun, and A. Meisels, "Efficient equilibria in a public goods game," in *2015 IEEE/WIC/ACM Int. Conf. Web Intell. and Intell. Agent Tech. (WI-IAT)*, vol. 2, 2015, pp. 214–219.

[25] S. Yu, K. Zhou, P. Brantingham, and Y. Vorobeychik, "Computing equilibria in binary networked public goods games," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, pp. 2310–2317, 2020.

[26] "TopCoder," https://www.topcoder.com/, 2020.

[27] A. Bernergård and K. Wärneryd, "Self-allocation in contests," Stockholm School of Econ., Tech. Rep. 2017:2, 2017.

[28] V. Bilò, L. Gourvès, and J. Monnot, "Project games," in *Proc. 11th Int. Conf. Algorithms and Complexity, CIAC 2019, Rome, Italy, May 27-29, 2019*, P. Heggernes, Ed., vol. 11485. Springer, 2019, pp. 75–86.

[29] S. McConnell, "What does 10x mean? Measuring variations in programmer productivity," in *Making Software - What Really Works, and Why We Believe It*, A. Oram and G. Wilson, Eds. O'Reilly, 2011, pp. 567–574.

[30] F. P. Brooks, *The Mythical Man-Month (anniversary ed.)*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[31] B. Vasilescu, K. Blincoe, Q. Xuan, C. Casalnuovo, D. E. Damian, P. T. Devanbu, and V. Filkov, "The sky is not the limit: Multitasking across GitHub projects," in *Proc. 38th Int. Conf. Softw. Eng., ICSE 2016, Austin, TX, USA, May 14-22, 2016*, L. K. Dillon, W. Visser, and L. Williams, Eds. ACM, 2016, pp. 994–1005.

[32] C. Wasser, "Incomplete information in rent-seeking contests," *Econ. Theory*, vol. 53, no. 1, pp. 239–268, 2013.

[33] B. Rogers, T. Palfrey, and C. F. Camerer, "Heterogeneous quantal response equilibrium and cognitive hierarchies," *Journal of Economic Theory*, vol. 144, no. 4, pp. 1440–1467, 2009.

[34] C. F. Camerer, T. Ho, and J.-K. Chong, "A cognitive hierarchy model of games," *Quarterly J. of Econ.*, vol. 119, no. 3, pp. 861–898, 2004.

[35] G. Cheikbossian, "The collective action problem: Within-group cooperation and between-group competition in a repeated rent-seeking game," *Games and Econ. Behavior*, vol. 74, no. 1, pp. 68 – 82, 2012.

[36] J. W. Boudreau and N. Shunda, "Tacit collusion in repeated contests with noise," *Conflict Studies: Scientific Study eJournal*, 2015.

[37] M. Serena, "Sequential contests revisited," *Public Choice*, vol. 173, no. 1-2, pp. 131–144, 2017.

[38] A. Nelson, "Deterrence in sequential contests: An experimental study," *J. Behavioral and Experimental Econ.*, p. 101541, 2020.

[39] V. Stray, N. B. Moe, and R. Hoda, "Autonomous agile teams: challenges and future directions for research," in *Proc. 19th Int. Conf. Agile Softw. Development, XP 2019, Companion, Porto, Portugal, May 21-25, 2018*, A. Aguiar, Ed. ACM, 2018, pp. 16:1–16:5.

[40] J. Morgan, D. Sisak, and F. Várdy, "The ponds dilemma," *The Econ. Journal*, 2017.

[41] M. Mavronicolas, I. Milchtaich, B. Monien, and K. Tiemann, "Congestion games with player-specific constants," in *Math. Found. of Comput. Sci. 2007*, L. Kučera and A. Kučera, Eds., 2007, pp. 633–644.

[42] E. Hasnain, T. Hall, and M. J. Shepperd, "Using experimental games to understand communication and trust in agile software teams," in *6th Int. Workshop Cooperative and Human Aspects of Softw. Eng., CHASE 2013, San Francisco, CA, USA, May 25, 2013*. IEEE Comput. Soc., 2013, pp. 117–120.

[43] N. Kitagawa, H. Hata, A. Ihara, K. Kogiso, and K. Matsumoto, "Code review participation: game theoretical modeling of reviewers in gerrit datasets," in *Proc. 9th Int. Workshop Cooperative and Human Aspects of Softw. Eng., CHASE@ICSE 2016, Austin, Texas, USA, May 16, 2016*. ACM, 2016, pp. 64–67.