Comparisons on Scrum Team Strategies: A multi-agent Simulation

Zhe Wang
Lincoln University
New Zealand
Zhe.wang@lincolnuni.ac.nz

ABSTRACT
Scrum is a type of agile process that incrementally, iteratively and continuously deliver software based on sprint time box. It is composed by User Stories, product backlog, sprint backlog, scrum team and sprints. Scrum team take user stories from product backlog into sprint backlog to start each sprint and deliver products at the end of each sprint. Sprint retrospective and review occurs at the end of each sprint to evaluate the delivered products and team performance. Based on the Scrum guide, scrum is easy to be understood but hard to be measured. Especially, it is depended largely on the performance of team dynamics referring to team compositions and task allocations, as its optimization make big impact on each sprint result. A new type of strategy called Intelligent pair strategies are tested in this paper to compare their performance under various task set and scrum team context.

CCS Concepts
•Computing methodologies~Modeling and simulation~Simulation types and techniques~Agent / discrete models

Keywords
Scrum, team dynamics, agent-based modelling, multi-agent system, team Strategies, solo programming, pair programming.

1. INTRODUCTION
Agile is a concept of creating software through iterative and incremental process. (Beck & Fowler, 2001) describes agile as "Individuals and interactions over processes and tools; Working software over comprehensive documentation; Customer collaboration over contract negotiation; Responding to change over following a plan." Scrum is one of the agile processes that realizes the agile manifesto where it splits user stories into several parts and aims to achieve part of them within a time period which can last from one day to thirty days as a sprint. Within each sprint, the process is like the waterfall model which is also composed of requirement analysis, system design, coding, testing and maintenance. However, there are more opportunities for scrum team and customer (product owner) to discuss on the software during the process because there are several sprints in an agile project and each sprint contains requirement analysis which may be refined at each sprint. Each sprint is able to deliver a higher project success rate compared to the waterfall approach because the size of software is smaller, its design goal is clearer, and the Scrum team is fully focused on the project. Scrum is composed by user stories, tasks, sprints, sprint meeting, deliverable software and Scrum team. User stories are designed based on user requirements. Each user story contains one or more tasks that need to be completed by the Scrum team in one sprint or several sprints. Scrum team will have daily sprint meeting to discuss what has been done and what needs to be done. Scrum team is composed of several members, which have different skills and capabilities, such as designer, developer and tester. Scrum team members need to interact and collaborate with each other to achieve the goal of the team incrementally or iteratively through sprints.

Although the Agile Scrum approach is an improvement over the waterfall model, it still suffers from several problems. One such problem is the team dynamics, which largely affect the quality, risk and value of the process. Team dynamics refers to team composition, task allocation, interactions between team members and how they work together. (Song et al., 2015) define effective team dynamics according to the following criteria indicated by (Nadler, Hackman & Lawler, 1979), there are:

- team performance (i.e., the product of teamwork meets the expectations of those who use it);
- member satisfaction (i.e., each team member's experience contributes to his or her personal well-being and development); and
- team adaptation (i.e., the team experience enhances each member's capability to work and learn together in the future)

A team consisting of experienced and highly skilled members will normally perform better than a junior team that is less experienced and skilled. In an Agile Scrum environment, the composition of a team will greatly affect the performance of the team, because the scrum team needs higher level of cooperation among the team members to achieve the sprint goal. Skills, experience and capabilities of the team members affect the performance of the team. Varying methods of tasks allocation may result in different outcomes and may affect the delivery of the software. It would be useful to investigate what kind of team dynamics leads to a more efficient and high-quality software delivery because team dynamics is affected by several factors such as capability, skills, roles and responsibilities and how well the members work together. Our study investigates how team composition, task allocations and team strategies can be used to improve the performance of the team in delivering a timely and high-quality software.

ICCMS '20, February 26–28, 2020, Brisbane, QLD, Australia

ACM ISBN 978-1-4503-7703-4/20/02…$15.00
DOI: https://doi.org/10.1145/3408066.3408087

2. REVIEW

2.1 Scrum

Software products can be very complex because of complexity in development, requirement analysis, technology adoption, functional complexity. Complexity mainly comes from the user requirements, which need to be addressed by the development team in conjunction with the software user who is the product owner.

SCRUM is one of the more popular agile methods which can deal with complex software system production. It is a software development process that was developed by Ken Schwaber and Jeff Sutherland in the United States which has been widely adopted and has become a common software development method (Schwaber & Sutherland, 2017)

SCRUM is defined as "A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value." This section describes Scrum and its processes in detail as shown in Figure 1.

2.2 Pair Programming

The pair programming(Arisholm, Gallis, Dyba, & Sjoberg, 2007; Bryant, Romero, & du Boulay, 2008; Chen, 2018; Cockburn & Williams, 2001; Coman, Robillard, Sillitti, & Succi, 2014; Dybå, Arisholm, Sjøberg, Hannay, & Shull, 2007; Gómez et al., 2017; Haider & Ali, 2011; Noori & Kazemifard, 2015; Z. Wang, 2018) task allocation for each task is much more complex than solo programming, because it needs to take two agents working on a task. Normally, the scrum master agent will still receive all the corresponding agents' preference value on the task list, and the scrum master agent should take the task which has the highest priority to be allocated first. The scrum master agent will choose the most appropriate pair for the task. Such as the complex task can be worked by expert-novice pair, expert-intermediate pair or intermediate-intermediate pair. The scrum master should choose expert-novice pair as the best choice as an example. If there are more than two experts available, the scrum master agent should compare those experts based on its preference value on the same task and pick the most appropriate expert. Then the scrum mater agent can pair the chosen expert with the novice to work on the task. For complex tasks, the scrum master agent should always get the high-level agent choose first than the low-level agent to form the pair, because without the high-level agent, this complex task will not be allocated even there are novice agent available.

3. STRATEGIES

3.1 Intelligent Solo and Pair Programming Strategies Design I (IPI)

3.1.1 Agent Decision Strategy

The agent decision strategy as shown in Table 1 is used to describe the agent self-control on what task it will task based on its own preference, such as novice will more like to work on easy task, intermediate will more like to work on intermediate task and expert will more like to work on complex task. The agent decision strategy will also service for its own benefit and motivation. However, such strategy would be better to take care all the member of the team, as novice would have more chance to get the easy task it can do, and expert will in charge of the complex task and be the right person to take the complex role. Intermediate agent also has more chance to get the intermediate task.

The pair agent choosing is different from task selection, it is totally based on the scrum master agent's decision making through the rules of the strategy. Such process is always happening after the first agent has already choose by the scrum master agent. The maximum benefit of pair is considerate for enhancing the pairing benefit, rather than any agent can pair with any agent. Such as the expert agent will not pair with another expert, as it may cause conflict, rather than benefit. Intermediate will not pair with intermediate to work on intermediate task, but two intermediates can pair to work on complex task. Novice is happy to pair with novice to work on easy task, However, we do not recommend to pair novice with intermediate or expert to work on easy task, because this is actually not well in the benefit of pairing, we would prefer to allocate intermediate or expert to work solo in easy task as well if there is no novice currently available.

Based on the literature of pairing programing, expert can lead intermediate or novice to work on complex task, and it's better for expert to lead novice to work on intermediate task as well, and we do not recommend using expert to lead intermediate to work on intermediate task, because this is waste of team resource and may cause conflict between the expert and the intermediate, because intermediate alone can work on the intermediate task. However, intermediate would be very happy to lead novice to work on the intermediate task.

Table 1. agent decision strategy in the intellgeint solo and pair

Agent	task	Best choice	Second choice	Third choice
Novice	easy	Novice-novice pair	Novice solo	no
	intermediate	Intermediate-novice pair	Expert-novice pair	no
	complex	Expert-novice pair	no	no
Intermediate	easy	Intermediate novice pair	Intermediate solo	no
	intermediate	Intermediate novice pair	Intermediate solo	no
	complex	Expert-intermediate pair	Intermediate-intermediate pair	no
expert	easy	Expert-novice pair	Expert solo	no
	intermediate	Expert-novice pair	Expert solo	no
	complex	Expert-intermediate pair	Expert-novice pair	Expert solo

3.1.2 Task Allocation (a)

The task allocation as shown in Table 2 is happened later than all agents has shown it preference on working tasks. The purpose of introduce task allocation after all agents has done its preferred selection is because this will makes the task allocation process more feasible and reliable, for example, if three novice want to work on an easy task, then the task allocation will first choose the most appropriate novice to be the first agent in doing the task and then help this novice agent to choose a partner from the other two novice to form a paired team. This is normally how agents shows it preference value and task get allocated in pair. Then scrum master agent will help each agent to choose the best partner based on the maximum benefit of pairing. It is not compulsory that each task must be worked by two agents, or each agent must be paired to work together, as this is an intelligent way of pairing, we will only allow pair that does not do any harmful. If a pair that may cause conflict, we will not set the pair to work and let the solo programming to go for the task. Pairing is always the first choice

for each task allocation, but this a pre-requirement is that the pair will not cause any conflict, otherwise solo is the choice. Those harmful pairing including expert with expert pair, expert with intermediate to work on intermediate/easy task, intermediate with intermediate to work on intermediate/easy task.

Table 2. task allocation(a) IP I TYPE A

Task	First choice	Second choice	third choice	forth choice
easy	Any Agent Novice	Any Agent Solo		
intermediate	Intermediate Novice	Intermediate solo	Expert Novice	Expert solo
complex	Expert Intermediate	expert Novice	Expert solo	Intermediate Intermediate

The task allocation and agent strategy can be formed into integrated together in the current algorithm design, based on preference-value for the first agent selection and maximum benefit of pair for the second agent selection. That selection can only be done by the system master agent, instead of the working agents themselves, as working agents themselves can only shows its preference value on task selection.

3.2 Algorithm Explanation on How Preference Value Support IP Strategy Implementation

The example shows tasks and agents distribution as Table 3. The agent and task have 1-10 levels, where level 1 means the easiest task and novice agent, while the level 10 means the most complex task and expert agent. 1-4 level are regards as novice and easy, 5-7 are regards as intermediate, 8-10 are regards as complex and expert.

Table 3. algorithm explanation

Agent level	1	2	3	4	5	6	7	8	9	10
	Agent1	Agent6	Agent4		Agent3		Agent7	Agent5	Agent2	Agent10
					Agent8					Agent9
Task complexity		Task 26	Task 16	Task46	Task 11	Task1	Task 31	Task 21	Task6	Task41

Firstly, each agent will select the task based on its own preference value on the task(Z. Wang, 2019).

For example, the task 46 is an easy task and it will be allocated to the best available novice as the first agent, then pair based on maximized the pairing benefit with another novice or working in solo.

For example, Task 31 is the intermediate task, those agents may send its preference value to take the task, there are all the expert agents and intermediate agents. In this situation, if agent 7 is available, then this agent is the best first agent. However, if agent 7 is not available, then the scrum master agent should choose agent 3 or agent 8 as the best first agent, if there is no intermediate agent available, then the scrum master agent will choose agent 5 or other expert. Intermediate agent has higher priority than expert agent on intermediate task allocation, even its preference value is negative, because that intermediate agent can do the intermediate task with just little lower capability. Whatever who get the intermediate task, then pair based on maximized the pairing benefit with another novice or working in solo.

For example, task 21 is the complex task, those agents may send its preference value to take the task, there are all the expert agent and intermediate agent. The scrum master agent should find the most perfect expert agent as the first agent, such as agent 5. If agent 5 is not available, then it should get another expert such as agent2. If expert get the complex task, then pair based on maximum the pairing benefit with another novice or intermediate or solo. However, if there is no expert, then the scrum master agent must find two best intermediate agents available before deciding who is the best first agent. Because a single intermediate cannot work on complex task.

3.3 Intelligent Solo and Pair programming Strategies Design II (IPII)

3.3.1 Task Allocation (a)

- Further adjusted Intelligent pair and more help low level novice and low-level intermediate agent
- Even negative agent (novice and intermediate) can be the leader of the team
- Negative novice can lead an easy task to pair
- Negative intermediate can lead an intermediate task to pair
- The new type of IP strategy shows in Table 4.

Table 4. task allocation IP II TYPE A

Task	First choice	Second choice	third choice	forth choice
easy	Novice Novice	Any Agent Novice	Any Agent Solo	
intermediate	NOVICE Intermediate	NOVICE EXPERT	Intermediate SOLO	EXPERT SOLO
complex	NOVICE EXPERT	Intermediate EXPERT	Expert solo	Intermediate Intermediate

4. EXPERIMENTS

All experiments is carried on based on the tool developed(Wang, Liu, & Wang, 2015; Z. Wang, 2018; Zhe Wang, 2018, 2019a, 2019b)

4.1 Idle Testing in IP (Intelligent solo and Pair programming) Strategy I and II

4.1.1 Testing Scope

Table 5 shows the team distribution and three different task sets, set I, II and III. The team is composed by 5 agents with various level as shown in the table 5. The task set is composed by 10 user stories, which each user story has 5 different tasks as shown in table 5.

Table 5. Team and Task set

1	2	3	4	5	6	7	8	9	10
		A4		A3	A1		A5	A2	

level	1	2	3	4	5	6	7	8	9	10
Set I		1			1	1		1		1
Set II		1	1	1				1		1
Set III				1	1	1	1	1		

4.1.2 Testing in Task Set I

In this testing case, I use the above 5 agents to work on the task set I for repeated 10 times.

Table 6. agent level and its idel time comparison

	Average Complete time	Average working time	Average idle time
IP I	**87.5**	**398.5**	**21**
IP II	**88.3**	**405.3**	**19.4**

4.1.3 Testing in Task Set II

In this testing case, I use the above 5 agents to work on the task set II for repeated 10 times.

Table 7. the agent compleiton time, working time and idle time

	Average Complete time	Average working time	Average idle time
IP I	**78.5**	**361.4**	**13**
IP II	**79.1**	**367.4**	**9.9**

4.1.4 Testing in Task Set III

In this testing case, I use the above 5 agents to work on the task set III for repeated 10 times.

Table 8. agent completion time, working time and idle time

	Average Complete time	Average working time	Average idle time
IP I	**81.3**	**369**	**22.9**
IP II	**82.7**	**390.3**	**8.3**

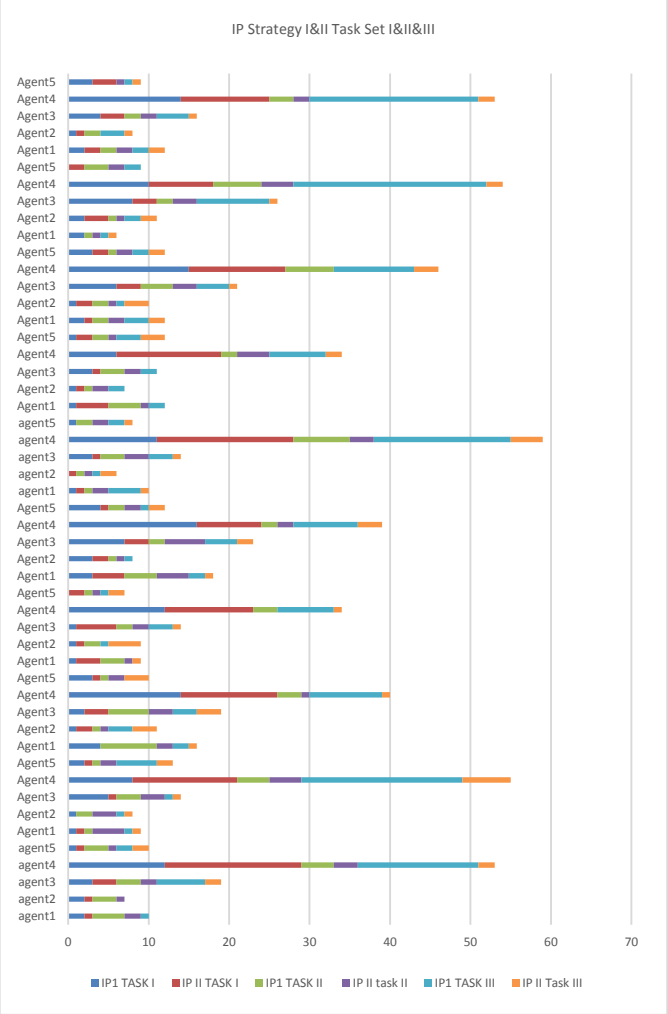

Figure 1. Agent Idle time in Each run of the 10 runs

5. SUMMARY

Figure 1 shows each agent's idle time accumulation during the three type of task set testing by using IPI or IPII strategy respectively. It can be observed that agent 4 and agent 3 has the highest accumulated idle time, this is because agent 4 is novice agent at level 3 and agent 3 is the lowest intermediate agent at level 5. Which further give us hints that the team working strategy design should considerate about the low-level agent, it is those agents that result the team idle time to be high.

Based on table 6 we can observe that IPI with completion time 87.5 is similar to IPII which is 88.3, their idle time 21(IPI) and 19.4(IPII) are also similar. In table 7 we can observe that IPI with completion time 78.5 and IPII with completion time 79.1, however the idle time of IPII (9.9) is lower than IPI (13). In table 8, the completion time of IPI and IPII are similar, however the idle time of IPII (8.3) is much lower than IPI (22.9). which gives us hints that the IPII do much better than IPI in idle time reduction.

6. ACKNOWLEDGMENTS

My thanks to Dr Patricia Anthony and Dr Stuart Charters at Lincoln University, New Zealand.

7. REFERENCES

[1] Arisholm, E., Gallis, H., Dyba, T., & Sjoberg, D. I. K. (2007). Evaluating Pair Programming with Respect to System

Complexity and Programmer Expertise. IEEE Transactions on Software Engineering, 33(2), 65-86. doi:10.1109/TSE.2007.17

[2] Beck, K., & Fowler, M. (2001). Planning Extreme Programming.

[3] Bryant, S., Romero, P., & du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. International Journal of Human-Computer Studies, 66(7), 519-529. doi:https://doi.org/10.1016/j.ijhcs.2007.03.005

[4] Chen, K. (2018). Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses. JISE, 29, 53-64.

[5] Cockburn, A., & Williams, L. (2001). The costs and benefits of pair programming. In Extreme programming examined (pp. 223-243): Addison-Wesley Longman Publishing Co., Inc.

[6] Coman, I. D., Robillard, P. N., Sillitti, A., & Succi, G. (2014). Cooperation, collaboration and pair-programming: Field studies on backup behavior. Journal of Systems and Software, 91, 124-134. doi:https://doi.org/10.1016/j.jss.2013.12.037

[7] Dybå, T., Arisholm, E., Sjøberg, D. I. K., Hannay, J. E., & Shull, F. (2007). Are Two Heads Better than One? On the Effectiveness of Pair Programming. IEEE Software, 24(6), 12-15. doi:10.1109/MS.2007.158

[8] Gómez, O. S., Aguileta, A. A., Aguilar, R. A., Ucán, J. P., Rosero, R. H., & Cortes-Verdin, K. (2017). An Empirical Study on the Impact of an IDE Tool Support in the Pair and Solo Programming. IEEE Access, 5, 9175-9187. doi:10.1109/ACCESS.2017.2701339

[9] Haider, M., & Ali, I. (2011). Evaluation of the Effects of Pair Programming on Performance and Social Practices in Distributed Software Development.

[10] Noori, F., & Kazemifard, M. (2015, 27-29 April 2015). Simulation of pair programming using multi-agent and MBTI personality model. Paper presented at the 2015 Sixth International Conference of Cognitive Science (ICCS).

[11] Schwaber, k., & Sutherland, J. (2017). The Scrum Guide.

[12] Song, H., Chien, A. T., Fisher, J., Martin, J., Peters, A. S., Hacker, K., . . . Singer, S. J. (2015). Development and Validation of the Primary Care Team Dynamics Survey. Health Services Research, 50(3), 897-921. doi:10.1111/1475-6773.12257

[13] Wang, L., Liu, W., & Wang, Z. (2015). The Implementation of Mini-Enterprise Business Process Management.

[14] Wang, Z. (2018, 23-25 Nov. 2018). The Impact of Expertise on Pair Programming Productivity in a Scrum Team: A Multi-Agent Simulation. Paper presented at the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS).

[15] Wang, Z. (2018). Teamworking Strategies of Scrum Team: A Multi-Agent based Simulation. Paper presented at the Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen, China.

[16] Wang, Z. (2019a). The Compare of Solo Programming Strategies in a Scrum Team: A Multi-agent Simulation Tool for Scrum Team Dynamics, Cham.

[17] Wang, Z. (2019b). Estimating Productivity in a Scrum team: A Multi-Agent Simulation. Paper presented at the Proceedings of the 11th International Conference on Computer Modeling and Simulation, North Rockhampton, QLD, Australia.

[18] Wang, Z. (2019, 18-20 Oct. 2019). P-value Based Task Allocation in a Scrum Team: A Multi-Agent Simulation. Paper presented at the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS).