AMERICAN UNIVERSITY OF BEIRUT

MODELING RESOURCE ALLOCATION AND INTERACTIONS IN COLLABORATIVE PRODUCT DEVELOPMENT USING AGENT BASED SIMULATION

by

MARK ADEL AL ACHKAR

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering Management
to the Department of Industrial Engineering and Engineering Management
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
February 2020

AMERICAN UNIVERSITY OF BEIRUT

MODELING RESOURCE ALLOCATION AND INTERACTIONS IN COLLABORATIVE PRODUCT DEVELOPMENT USING AGENT BASED SIMULATION

by
MARK AL ACHKAR

Approved by:

[illegible]

Dr. Ali Yassine, Full Professor — Advisor
Department of Industrial Engineering and Management

[illegible]

Dr. Saif Al-Qaisi, Assistant Professor — Member of Committee
Department of Industrial Engineering and Management

[illegible]

Dr. Hiam Khoury, Associate Professor — Member of Committee
Department of Civil and Environmental Engineering

Date of thesis/dissertation defense: February 17, 2020

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Al Akkar Mark Adel
Last First Middle

[✓ Master's Thesis] Master's Project Doctoral Dissertation

☐ I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☒ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes
after : **One -X- year from the date of submission of my thesis, dissertation, or project.**
Two ---- years from the date of submission of my thesis, dissertation, or project.
Three ---- years from the date of submission of my thesis, dissertation, or project.

[illegible] 20/02/2020

Signature Date

ACKNOWLEDGMENTS

Special thanks to Dr. Ali Yassine who provided continuous guidance and advice from the start until the end of this research for over two years. I would also like to thank Dr. Hiam Khoury and Dr. Saif Al-Qaisi for being very supportive committee members, aiding and advising me until the last moment.

My gratitude goes to the American University of Beirut and specifically the Department of Industrial Engineering and Engineering Management which helped come a long way and insured the suitable environment for my growth.

A final thanks to all my friends, colleagues and family members who helped me complete this research and contributed to my personal development.

ABSTRACT

AN ABSTRACT OF THE THESIS OF

Mark Adel Al Achkar for Master of Engineering Management

Title: Modelling resource allocation and interactions in collaborative product development using Agent Based Simulation

The main objective of the proposed research is to study collaborative and distributed development projects. The availability of resources, their properties (i.e., skills, learning-ability, etc.), their improvement (i.e. learning, advancing…), and their interaction dynamics (i.e., predecessor relationships, meetings, etc.) play a major role in the allocation and optimization of resources in such an environment (i.e., collaborative and distributed development environment). Using Agent-Based Simulation, the objective of this research is to estimate and reduce the effort and duration for any product development project using an innovative resource allocation strategy that considers the agents' technical and managerial skills, their overall experience, and their learning ability. By simply adjusting the Agent-Based Simulation model's configuration and settings, the model presents itself as a flexible tool to simulate the product development process and covers different approaches from the traditional methodologies such as the waterfall model, to the more recent PD processes like the Agile model. The model implemented was able to successfully demonstrate the usefulness of each of its features in influencing development time and effort. For example, it shows improvement in the development time when using the proposed resource allocation strategy, a decrease in the project time when agents cooperate on tasks, demonstrates the effect of learning on task execution, and the impact of rework on the PD process. Additionally, two case studies were introduced. The case studies show the flexibility of the model and the difference between the Agile model and the waterfall model, from an effort and time perspective. Finally, the model was validated using five different projects from the software development industry. Each of the project's managers provided feedback and validated the output of the model. The model was successful in simulating several real-life projects related to the software development industry and outputting accurate results.

CONTENTS

ACKNOWLEDGMENTS V

ABSTRACT VI

INTRODUCTION 1

BACKGROUND 5

2.1 Product Development Environment 5

2.1.1 Task Dependency 5

2.1.2 Task Allocation Methods 9

2.1.3 Agent Interactions 13

2.1.4 Rework/Iteration 21

2.1.5 Employee Learning 22

2.2 Simulation Techniques 24

2.3 Existing Simulation Models for Product Development Processes 28

2.4 Agile and Scrum Project Management 31

AGENT-BASED MODEL FOR RESOURCE ALLOCATION & INTERACTION IN PD 34

3.1 **Task Allocation Method** 35

3.2 Meetings & Collaboration 39

3.3 Cooperation 41

3.4 Agent Learning 42

3.5 Task Dependency 43

3.6 Implementation 44

3.6.1 Agent and tasks creation 45

3.6.2 Task Allocation 46

3.6.3 Simulation Process 47

3.6.4 Meetings & Rework 49

3.6.5 Idle Time 52

3.6.6 Process Flowchart 52

TESTING, ANALYSIS, AND DISCUSSION 56

4.1 SDC Task Allocation 59

4.2 Agent Learning 62

4.3 Dependency Priority 65

4.4 Cooperation 66

4.5 Client Meeting 69

4.6 Team Meetings 72

CASE STUDY 74

5.1 Agile Model 74

5.2 Waterfall Model 80

VALIDATION 84

CONCLUSION 92

REFERENCES 95

APPENDIX A 101

APPENDIX B 102

APPENDIX C 103

CHAPTER 1

INTRODUCTION

The optimal distribution of development resources is a central issue in managing the product development (PD) process. However, the current emphasis on how to handle and use these resources has reached peak levels with industries seeking to not only lower their costs and maintain their product quality but also to attract and develop talented employees. PD is defined as the set of activities starting with a market opportunity and ending with production, sale, and product delivery (Ulrich & Eppinger,2012). The goal of the PD process is to create products that fulfill the customers' needs. Many researchers have focused on the importance of task allocation in the PD process, whether it is software or a hardware product. However, previous literature studied separately task allocation, cooperation, collaboration, meetings, employee improvement, as well as task dependency, and how each affects the PD process. This paper will focus on combining all of these features in a single agent-based simulation model that can study the effect of these individual features – as well as their interaction- on the PD process. So, the problem is PD projects take longer, cost more, and resources are not well utilized and assigned throughout the project. This could be due to not having enough meetings to coordinate the various work packages, having too many wasteful meetings, misuse of scarce resources - specifically human resources, as well as failed negotiation mechanisms. Many models have been proposed to study this PD environment, but they have been specific to one or two of the mentioned features and how they affect the PD environment.

Since this thesis investigates multiple stages or phases of agent interactions, a few definitions are necessary. We shall distinguish between five modes of interactions: Negotiation, Communication, coordination, collaboration, and cooperation. We define Negotiation as the way for conflict resolution. In any conflict, the opposing sides would have to negotiate and compromise to reach a mutually agreeable decision. Communication includes all the different methods for sharing information and knowledge among the agents or between the agents and the client. Coordination is defined as the managerial aspect of dividing tasks between agents. Collaboration is work done by multiple agents together to achieve or create something not entirely feasible by one agent alone – collaboration indicates a creative aspect of working together. Cooperation is having multiple agents working on the same task – physically or side by side - in order to complete it faster. Within any PD team, the need for all these five interaction modes is almost obligatory.

In particular, software projects have high percentages of failures due to many reasons. Incomplete or evolving requirements, lack of resources, unrealistic expectations are some of the causes of delivery failure (Kusumasari et al. , 2011). These reasons are mainly attributed to lack of communication among the developers or employees and the users or clients. (Moenaert et al. ,2000). Other reasons for project failures also exist and they are not restricted to software projects. Pinto and Mantel (1990) defined several factors that affect project implementation and could impact the success of the project. Out of these factors, we mention client's feedback and monitoring, the project schedule, and the personnel. Within any project, there exists the possibility of incorrect task allocation, poor soft skills, unneeded meetings, etc. which could all be the root cause of project failure. Dinu (2016) discussed project failures and identified several reasons. Some of these reasons include poor planning, inaccurate cost estimation, lack of communication, and failure to set expectations. Additionally, the probability of rework due to client changes or adaptations always causes the project to be late. The initial estimates for tasks within a project do not always consider the rework by the client. Thus, arises the need for proper management tools and techniques for resource allocation, employee learning and improvement, meetings, and cooperation, all of which can estimate how the project should be managed. These features also provide a way to estimate the project's chances of success. By improving coordination and meetings, the project is rendered easier (Dingsoyr et al. , 2018). By working on the allocation techniques, estimating employee learning, and indicating the need for cooperation, the project has a higher chance of success. These mentioned factors would also affect the project's budget and cost, as well as scheduled time.

Employee improvement is the possibility of any employee improving his/her skills and ability due to learning by doing. This is another factor that comes to play in the PD process. By working on the same type of tasks or learning new skills by completing different types of tasks, the employee/agent should be improving eventually.

Another factor to consider while focusing on task allocation is task dependency. Some tasks may require prerequisite tasks to be completed in order for it to start. This happens on a regular basis in the PD Process and is reflected via task availability – as the dependent task would not be available until its prerequisite task is completed. Another complicating factor that we consider in this study is the project management approach, such as waterfall/sequential or Agile/Scrum. Each of these approaches has its own characteristics and specifications which are discussed in the coming chapters.

Agent-based models (ABM) can explicitly model the complexity arising from individual actions and interactions that arise in the real world. In other words, ABM provide the capability to model real-world systems of interest in ways that were either not possible or not readily accommodated using traditional modeling techniques (Siebers et al., 2010). A simple example would be to consider a driving intersection. Each driver arriving at the intersection is an independent agent. The driver/agent would then have to take the decision of which way to go and by doing so, multiple drivers would interact with each other and with the surrounding environment. Their interactions with each other could be as simple as a driver letting another pass or not. Using Netlogo, a powerful agent-based simulator, we are able to see the effectiveness of ABM as each simulated employee will choose their task based on the various simulation options set by the user. This will allow us to test multiple scenarios and use cases to study the impact of each mentioned factor in the process.

The goal of this research is to find an improved resource allocation technique for different Product Development (PD) processes while taking into consideration the different aspects within project management represented by employee interactions, learning and improving, meetings, rework, and task dependency.

In the rest of this document, we will go through the existing literature related to the product development environment and different simulation techniques in chapter 2. Then, we will discuss the proposed model in chapter 3. Having covered the model, the next step would be to perform sensitivity testing for the features in chapter 4 and see how each feature affects the model. Chapter 5 includes the two case studies considered in this paper. Chapter 6 is the validation of the model based on real life projects. And finally, we conclude in chapter 7.

CHAPTER 2

BACKGROUND

The background section is divided into the following subsections: (1) PD Environment, (2) simulation techniques, (3) existing PD simulation models in the literature, and (4) Agile/Scrum project management.

2.1 Product Development Environment

This section discusses the PD environment and explains the process based on five typical scenarios: Task Dependency, Task Allocation Methods, Agent Interactions, Rework/Iteration, Employee Learning. Each of these scenarios effects the PD environment in its own way. In each scenario, specific literature is studied and analyzed to demonstrate its importance and effect on the PD process.

2.1.1 Task Dependency

When defining task dependency, the first thing to come to mind is that one task is dependent on another task or a group of tasks. But looking into it a bit more deeply would reveal that one task could be blocked due to another's incompletion. It could need a certain percentage of the predecessor task to be completed. To explain dependency and interdependency in a simple example, let us consider a car that is manufactured and then assembled at the factory. The manufacturing process must happen – and be completed – before the assembly could begin. This is a simple example that looks into the progress as two main processes, but what if the car's interior is fabricated and then assembled, before the rest of the manufacturing could continue; in which the task of manufacturing has a partial completion that allows the dependent task of assembling to start. Yet, when the assembly is partially completed, the manufacturing starts again.

To model task dependency, we first introduce the Design Structure Matrix (DSM). The DSM is a representation and analysis tool for modeling relationships between multiple components within a system. These relationships could be independent, dependent, or interdependent design tasks. The DSM is frequently used in the modeling and analysis of a typical PD process due to its ability to represent task interdependencies (Abdelsalam and Bao, 2007). Basically, it is a simple binary matrix (containing either 0 or 1 in each cell) to indicate for each column/row combination if the activity in the column is related to the activity in the row. The major advantage of the DSM over any diagraph is its compactness and the systematic mapping (Yassine et al., 2001).

Panchal (2009) discussed the importance of task dependency and its effect on the PD cycle. He proposed a model in which he considered the interdependencies among different modules of the system. Interdependency signifies that if a module A is dependent on module B – or has a relation with module B, this does not stop module B from also being dependent on module A. This could be simply reflected as a need for the module to have a percentage of another module completed to be available to be worked upon. This is handled by considering the dependency strength.

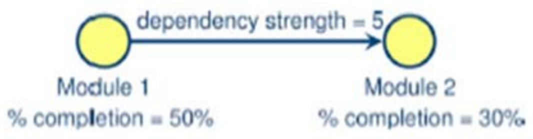


Figure 2-1 Dependency Strength (Panchal, 2009)

To explain the dependency strength, consider Figure 2-1. In this figure, having a dependency strength of 5 indicates that any change in Module 1 will result in a 5% rework in Module 2. This could be quite accurate if you consider that the changes in Module 1 will actually affect Module 2 since it is dependent on Module 1. Panchal (2009) also considered an example in which multiple modules are interdependent; the modules are split into 2 types: the core modules (modules 0-2) and the external modules (modules 3-8) as shown in Figure 2-2

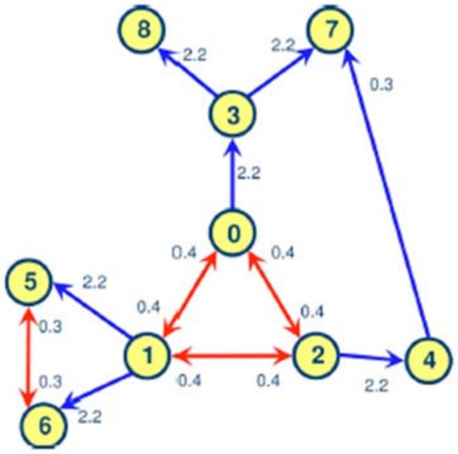


Figure 0-2 Interdependency (Panchal, 2009)

Figure 2-2 indicates that the core modules are basically independent with a very low dependency strength. On the other hand, having a strength of 2.2 between Module 2 and Module 4 signifies that for any change in Module 2, 2.2% of rework is required in Module 4.

According to Hoegl and Weinkauf (2005) interdependencies in a modular system require a lot of information exchange among the participating teams. They also stated that the success of one team is dependent on how good the team’s work integrates with other related teams’ works.

Basically, this indicates the importance of communication and coordination – discussed in Section 2.1.3, especially in a multi-team discipline.

Stobrawa et al. (2018) stated that the first step to build a model is to find interdependencies between workstations since the focus is on the whole system initially. Based on Figure 2-3, Stobrawa el al. (2018) considered the employees and their competences as inputs to the system along with the material information to obtain material output as additional information for the PD process.

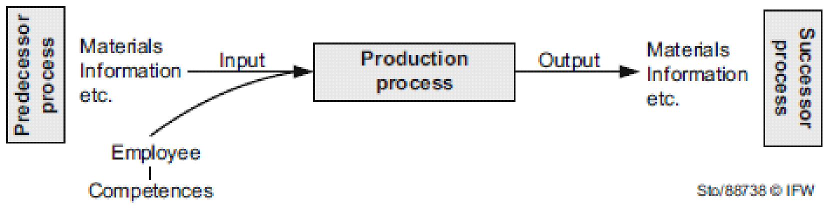
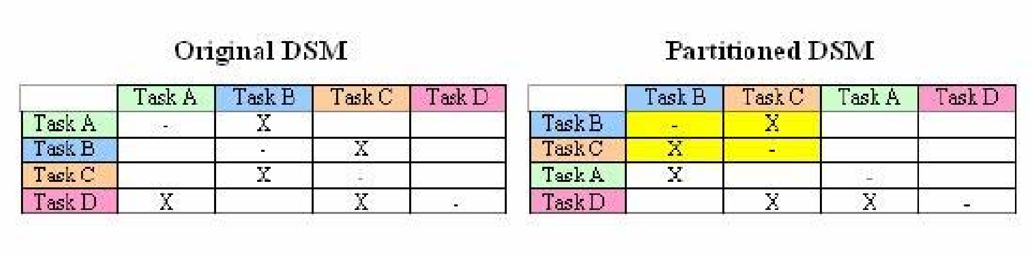


Figure 0-3 Production Process (Stobrawa et al. - 2018)

Johnson et al. (2008) also stated that due to task interdependencies, their DSM – studied in Section 2.1.2 - is transformed into blocks where the interdependent tasks are isolated in a triangular form and are transformed into the following Tables 2-1 and Table 2-2. In this case, Tasks B and C are interdependent (B depends on C and C depends on B) and are partitioned into their own block.



Original DSM

	Task A	Task B	Task C	Task D
Task A	-	X		
Task B		-	X	
Task C		X	-	
Task D	X		X	-

Partitioned DSM

	Task B	Task C	Task A	Task D
Task B	-	X		
Task C	X	-		
Task A	X		-	
Task D		X	X	-

Tables 2-1 and 2-2 Original & Partitioned DSM (Johnson et al., 2008)

2.1.2 Task Allocation Methods

Task allocation is the process of choosing which agent will be performing which task – based on certain criteria. Task allocation methods are an integral part of this research so this section will look into existing techniques and methods. To tackle this challenge, Salhieh and Monplaisir (2003) analyzed the resources available. The first resource they considered is the "People" resource. This resource is defined as actors with specific knowledge and skills (Technical, interpersonal, and decision-making). This would lead to the creation of Actor-Skill Vector (ASV). The ASV will set a value for each of the actor's skill in a direction, such as "Actor 1" would have a vector with as many directions as he/she has skills. This is reflected in the following vector formula (Salhieh and Monplaisir, 2003):

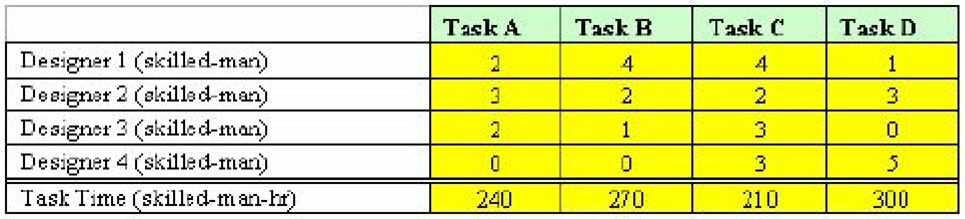
$$ASV(A_1) = \{k * S_1, k * S_2, \ldots, k * S_j\} \qquad \text{Equation(1)}$$

In Equation 1, 'j' is the number of skills A_1has, 'S_j' is the value of each skill, and 'k' is simply whether the actor has the skill or not ($k = 1$ if the actor has the skill, $k = 0$ otherwise). Consider that Actor 1 as a set of skills $S\{S_1 = 3, S_3 = 5\}$ (3 and 5 being arbitrary values for the skills). k would be 1 for $S1, S3$ only and 0 otherwise. In this case, for j skills, the Actor Skill Vector for Agent 1 would be: $ASV(A_1)$ = {3,0,5, …,0}.

The authors then focused on resource assignment which is to find the best way to choose which actors will perform which tasks. Basically, it starts with assigning actors to tasks based on skills. However, some constraints exist such as the cost of the actor, and its availability. To solve this, three strategies are presented: Obtaining new resources, re-sequencing the development process, and decoupling tasks. To create the assignment module, it is first necessary to allocate enough resources for the completion of the development process. Then, finding the constraints that limit development progress; such as cost, availability and efficiency. Finally, to assign resources to tasks, a 3D Matrix is formed consisting of task-skill-actor where a binary determination is used to set if skill has a matching actor or not. This approach for task allocation is similar to what is proposed in our model. However, the use of vectors for the number of skills was not needed as a different consideration was used. The proposed task allocation technique will be discussed thoroughly in Chapter 3.

The same issue is tackled by Johnson et al. (2008) by introducing a computational method for task allocation. This method is based on a specific managerial mindset, which proposes initially the following rules for the manager: focusing on the product's creation in an efficient, time-saving and minimal cost way, defining sets of tasks to perform, managing groups of design teams to complete the task. Johnson et al. (2008) used a DSM to partition the tasks.
With geographical constraints and challenges, it is difficult to determine the design teams. However, to overcome these constraints, Johnson et al. (2008) indicated that the focus is on the designers' competences and the forming of the designing team based on the assigned task. It is also possible to work on independent tasks sequentially by various design teams. To determine the designers for each task, the designer is rated on a 6-point scale (0 being no ability and 5 being an expert). The rating would also indicate other characteristics such as dedication and ambition.

Another criterion is the estimated time for each task to be completed by each designer which is usually done by experts and resulting in Table 2-3.



	Task A	Task B	Task C	Task D
Designer 1 (skilled-man)	2	4	4	1
Designer 2 (skilled-man)	3	2	2	3
Designer 3 (skilled-man)	2	1	3	0
Designer 4 (skilled-man)	0	0	3	5
Task Time (skilled-man-hr)	240	270	210	300

Table 2-3 Designer & Task Time (Johnson et al. ,2008)

Based on Table 2-3, the team is formed to complete the task in the least amount of time.
The team assignment is then automated using a Generic Algorithm (GA). As the purpose of the method is to optimize task allocation, this GA takes the Designer-Task matrix and task times to find the best team selection.

Acuña and Juristo (2004) also worked on the developer's personality as key criteria to assign roles. They created a software process model that handles the human capabilities factor. Their model consists of two relationships that are emphasized in the assignments. First is the capability-person relationship**:** matching the capabilities needed for a software project with the person's personality factors. The second one is a capability-role relationship. The aim here is to match the capabilities with the roles of a person in a software team, ranking them between high and medium depending on how critical the capability for the given role is. Finally, to assign people to roles, four steps are in order: comparison, evaluation, monitoring and consolidation, and finally documentation. The aim is to analyze each personal profile against each role to match them.

Acuña et al. (2006) discussed how to assess the team members' capabilities and assign them to roles that match them, while also matching the tasks' needs to the roles and their requirements. The first step is to emphasize the importance of assigning people to their most suitable role by identifying the individuals and their capabilities. The authors start off with defining the traits and personalities and continue to find the person's capabilities and skills. They suggested doing famous personalities tests, such as the NEO Personality Inventory-Revised test which addresses the "Big Five personality factors (extroversion, agreeableness, conscientiousness, emotional stability, and openness to experience)". Another common test mentioned by Acuña et al. (2006) is the 16 Personality Factors (16PF), a more detailed personality test that predicts people's behavior by studying people's mental health such as level of anxiety, adjustment, as well as emotional stability. Doing these personalities tests allowed them to assess the roles suitable for each employee and thus finding more suitable tasks for each.

The three competences considered by Stobrawa et .al (2018) for task allocation were: Performance, Experience, Level of Practice. Performance summarizes the employee's time and quality of work. Experience is how much experience the employee has in a specific work process. Assuming that more experience leads to better performance. Level of practice would be the fact that repeating the same activity should lead the employee to performing it better. The matrix is built based on these three competences, where each has a value between 0 and 1. The values are determined by comparing to a default value; if an employee needs twice the default time to complete a job, the performance would be 0, if the default time is respected the performance would be 1. A special algorithm is applied to determine the best coefficient for each employee to determine who should get the task. This coefficient is called the allocation score $a_{e,t}$, calculated as indicated in Equation 2:

$$a_{e,t} = \frac{\sum w_c . n_{c,e}}{\sum w_c} \qquad \text{Equation (2)}$$

In Equation 2, w_c is the weight for each competence,e is the total number of employees and t are the tasks considered, and n is the normalized value for each competence. These weights are different for each company and differ based on the focus of the considered company. Let us consider we are allocating task 1 where the number of employees e is 2, and there exist two competences with their respective weights $w_1 = 4$ and $w_2 = 2$. For employee 1, the normalized value for competence 1 is $n_{1,1} = 0.5$, and for competence 2, $n_{2,1} = 0.3$. For employee 2, $n_{1,2} = 0.4$ and $n_{2,2} = 0.8$

Then $a_{1,1} = \frac{w_1 * n_{1,1} + w_2 * n_{2,1}}{w_1 + w_2} = \frac{2.6}{6} = 0.43$ and $a_{2,1} = \frac{3.2}{6} = 0.53$. Based on these calculations of $a_{e,t}$, we would decide to allocate employee 2 for task 1 since its allocation score is higher. This basically indicates that its skills are closed to what the task requires.

Additionally, this topic was tackled by Kandemir and Handley (2019). They addressed the general assignment problem (GAP) which involves finding the minimum cost of assigning tasks to agents such that each task is assigned to one agent.

$$\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \qquad \text{Equation (3)}$$

Equation 3 represents the simulation optimization formula for the GAP. c_{ij} is the cost of having agent i on task j, x_{ij} is 1 if i performs j and is 0 otherwise. The goal is then to minimize this function in such a way that each task is assigned to only one agent while also considering the availability of each agent.

2.1.3 Agent Interactions

This section discusses the five types of agent interactions mentioned previously and summarized in Table 2-4.

Interaction	**Definition**
Negotiation	Interaction between opposing agents to resolve a conflict or any difference of views.
Communication	Vocabular interaction between the agents where discussions happen, and knowledge and information are shared
Coordination	Distribution and assigning tasks, while considering dependent work among the assignees.
Collaboration	Agents working together on creating or designing. (i.e. Brainstorming and creative thinking)
Cooperation	Agents working on performing tasks together at the same time.

Table 2-4. Agent Interactions Definitions

2.1.3.1 Communication

Communication is the first and most important type of interaction and so it is the most challenging. Moenaert et al. (2000), considered that for the success of any PD project – especially on a global scale, effectiveness in communication is a prerequisite. Allen (2007) concluded that that lack of communication among the team members has led to project failures. He also distinguished between three types of communication as described in Table 2-5.

Classification	**Description**
Type I	Communication to coordinate the work (Coordination)

Type II	Communication to maintain staff knowledge of new developments in their areas of specialization (Information)
Type III	Communication to promote creativity (Inspiration)

Table 2-5. Types of Technical Communication (Allen, 2007)

The definitions set by Allen (2007) in Table 2-5 are very similar to what we defined in Table 2-4. What Allen (2007) considered as types of technical communications and labeled Coordination, Information, and Inspiration respectively, is what Table 2-4 indicated to be Coordination, Communication, and Collaboration.

2.1.3.2 Coordination

Dingsoyr et al. (2018) discussed coordination in Agile Projects and concluded that Complexity in Agile projects makes it harder for their management; this complexity is due to the Agile flow, which works in differently spanned iterations that focus on completing specific features each time. IT Projects are more challenging for the managers to coordinate and assign tasks. Here lies the difference between traditional project management -such as the waterfall model - and agile management. The waterfall is always progressing downwards. Phases are executed in order, one after the other; a phase cannot start unless the previous is done (Bassil, 2012) [1]. On the other hand, agile tends to focus on self-management with processes and iterations being the goals. Another notable difference is the level of coordination and communication between the project

[1] The waterfall model goes through five phases: First phase is the analysis, in which all the project's requirements are acquired and studied. Second is the design, where the designers mainly focus on planning and solving the problem. Phase three is implementation, the realization of the project requirements. Fourth step is testing is the validation phase in which the systems are tested to meet the requirements. Final phase is maintenance, this is the post-delivery process to refine the deliverables (Bassil, 2012)

handlers and the stakeholders due to frequent releases. Additionally, Dingsoyr et al. (2018) discussed the importance of physical meetings since they allow richer communication for better coordination. Even though communication could be done by different methods such as virtual meetings, emails, and online chatting, physical meetings will push towards a better understanding among the attendees. A higher rate of group meetings set by the management usually leads to a technological novelty. In Agile specifically, this higher rate is insured through the iteration planning meeting (sprint meetings), daily meetings, as well as retrospectives. The daily meeting is called a daily scrum, a short meeting in which all the team members report in about their last day's activity and their current day planned activity. Agile project management is further discussed and explained in Section 2.4.

2.1.3.3 Collaboration

According to Kvan (2000), the term ‘Collaborative Systems’ mainly describes the use of multiple distant systems to support work between designers/developers. Computer-supported collaborative design is a term used frequently. This collaboration is key between designers to share information and data, as well as to provide communication. Kvan (2000) suggested that Collaboration success is achieved when a group can complete something undoable by individual, mainly in a creativity related aspect. Despite some implications that the maximum number of collaborators could only be around 4 or 5, studies have shown that the actual number has no real limit (Kvan, 2000). Collaboration in design would be different than another type of collaboration, since it needs a higher sense of work to achieve creativity. Collaboration Design is in general an act that involves multiple people. It could be a close coupled design process in which multiple designers work together to complete a single design product where their work is simply added together to complete the design product. In fact, most design work is in a loosecoupled process where each designer contributes to multiple design products with their respective knowledge and expertise.

Bergner et al. (2016) also discussed how collaborative activity across multiple grade levels demonstrated positive outcomes on different aspects. Organizational research has also shown the effectiveness of collaboration in improving performance in organizations; in an input-output framework, affected by communication and coordination, it is more likely to have effective team outcome. Additionally, Bergner et al. (2016) described personality and agreeableness - the trait of accepting others' opinions- and how a team with higher average agreeableness would exhibit better performance, primarily due to the coordination happening among the team members.

To relate the mentioned interaction modes to the types of meetings possible, let us consider that all types of interactions could occur in a physical meeting. Communication and negotiation would be easier during physical meetings due to the high level of interaction present among the attendees. Coordination could be done through virtual meetings due to the lack of physical exchange and information sharing; it is rather essential to simply distribute and coordinate the tasks and activities. Collaboration and cooperation could be done via physical, telephony, or internet-based communication methods. Since their purpose is working together either to find a solution or to implement one, physical attendance could provide an efficiency boost.

Johnson et al. (2008) suggested that by having regular meetings, design teams check what other teams' parameters are, which makes it easier for a conversion among them. However, there is an allowable separation – the distance between the teams' parameters, which informs the agents/teams how close or far they are from the others' parameters. The agent could be: In the allowable envelope, on the same side of the allowable reason as it was last meeting, or on opposite of the allowable reason as it was last meeting. Thus, there exists a meeting cost which is the amount of time spent to prepare and hold a meeting. So, to find the correct number of meetings, Johnson et al. (2008) proposed a convergence model. This model shows that after multiple product design iterations, the product design team should have converged to a solution. To prove this, multiple simulations are tested for the convergence time with different coordination rules - that would affect the level of coordination - such as having high frequency of meetings with short meeting times, or low meeting frequency with longer meeting times. These rules show that convergence is always easier to reach with coordination. Johnson et al. (2008) showed the difference between having too few, too many, and just enough meetings using different scenarios and frequencies and proved that with different meeting frequencies and meeting cost, the convergence time also varies. For example, Johnson et al. (2008) calculated the average for 1000 simulations and then compared the distributions of having a meeting for every 100,200, and 300 iterations in the system. Then, they compared the average convergence time. This model actually gives an insight on the importance of meeting frequencies and their durations to determine the best convergence among the team members.

We shall specifically see how meeting frequency, duration, and result will affect the PD process in Chapter 3.

2.1.3.4 Cooperation

Many researchers visited the definition of cooperation, and how it would affect the PD process. Leithold et al. (2016) discussed how cooperation could happen in small firms to encourage and improve the PD process. They indicated that for small firms to grow, and due to their limited financial and human resources, it would be easier for them to cooperate among each other. Similarly, Ernst et al. (2010) identified cooperation as a basic need for the PD process. However, their discussion focused on cooperation among different PD departments (Sales, Marketing, Research and Development). Thus, they focused on Cross- Functional Cooperation which indicates a certain level of involvement and information sharing between agents of the three mentioned departments. Olfati-Saber et al. (2007) presented the cooperation concept in networked multi-agent systems. They studied consensus and how it affects a multi-agent system, while also emphasizing the importance of cooperation. In their paper, Olfati-Sabet et al. (2007) interpreted cooperation as “giving consent to providing one’s state and following a common protocol that serves the group objective”.

2.1.3.5 Negotiations

The last defined interaction is negotiations. Kusiak and Wang (1994) discussed negotiation specifically and considered that nowadays concurrent design aims to ease and solve multiple constraints and challenges in a product design cycle as well as reducing cost and time. This would involve specialists of multiple backgrounds with limited knowledge of other disciplines that could cause conflicts among them or could help them improve. Thus, rises the need for negotiation. Conflicting goals among the specialists will need negotiations leading to a mutually accepted product design. Usually negotiations focus on finding a solution for the conflicts even in the most complex of processes. Kusiak And Wang (1994) developed a goal-directed negotiation model based on decision analysis which is proposed for resolving conflicts. This is done through a model that generates, analyzes and chooses the solutions based on three criteria: maximum joint utility, minimum individual utility differences, and a combination of these two. The design negotiation scenario is illustrated with an example of a valve. This valve will allow a type of fluid to pass at a certain pressure and will hold the fluid in equilibrium.

This example helps to imagine a design by three agents, where each spring and enclosure represents an agent. They must work together at a certain threshold and even compromise together to allow the fluid/project design to be achieved. This simple example represents how negotiation should happen to reach an equilibrium through coordination and perhaps compromise.

Scott and Antonsson (1996) considered that by formalizing the negotiation process, design teams can promote a better information exchange and trace the design history. Having a tool for formal negotiation would make it easier to include important performance goals. The authors dived into examples of why design negotiation is needed. One of the examples is having an incremental improvement for an existing automobile chassis to increase a certain attribute by 10%. If the task was not attained, the designers/engineers would begin a negotiation process. They could ask for additional resources or a compromise on the target. Here, negotiation serves as a method to discover and discuss the constraints faced.

Chien et al. (2013) separated between cooperative negotiation and competitive negotiation. They described the cooperative negotiation as the one that involves cooperation between various system levels through information sharing among the systems and components. Whereas the competitive negotiation is based on individual decision-making at a single level within the system. In their model, Chien et al. (2013) discussed a decision-making process to utilize both types of negotiations through different scoring methods for each to determine the decisions. In their research, Cooper and Taleb-Bendiab (1998) also discussed how a negotiation support should work in a multi-agent system. They suggested seven steps that vary from conflict detection and identification, to negotiation management and finally conflict solution. Thus, a model was created that handles the design agents, analyzes the designs suggested, and suggests their decision resolution through a Decision Tree where each arc is a decision.

2.1.4 Rework/Iteration

In any PD process, the probability of having to work on completed tasks could vary but is always present. Rework usually happens due to one of two reasons: Either it is due to a feedback from the client, or a change in a certain task leads to change in a dependent task. Browning and Eppinger (2002) suggested that since work is being repeated and the designer already knows that task's details, its rework duration would not be as long as the initial duration. They introduced DSMs (discussed in Section 2.1.1) that focus on the rework probability as well as the rework impact.

Yassine et al. (2001) discussed the probability of rework through the following steps. First step is to setup the subjective assessment by defining the Information Variability (IV) and Task Sensitivity (TS). IV describes the probability that the information regarding a certain task has changed, thus the task as whole could have been altered. TS on the other hand, describes how sensitive a certain dependent task is to the information change due to IV. Second step is mapping and calibration. This step mainly focuses on calibrating the rework scale, to see how much rework would have to be applied on the dependent task due to its sensitivity. Third and final step is validation. This is done by validating that the changes that caused the rework didn't ruin the system.

Simpeh et al (2015) built a model to determine the probability of rework and its distribution. By using a cumulative distribution function (CDF) and sample research data in the construction industry, they were able to create a table that specifies the probability of a certain percentage of rework required. For example, they determined that the mean total rework cost (as a percentage of the original cost) is 5.12% with a 76% that a project would exceed this rework cost.

2.1.5 Employee Learning

Employee learning is the ability and capacity for an employee to be learning new skills or improving existing ones. Whether by repeating the same type of tasks or by performing new types, the employee should have the ability to learn and improve their own skills – skill levels precisely, in a way that reflects them improving on the job. Learning could be considered as work experience as it would make sense that an employee who has a lot of experience is someone who has learned a lot and improved their skills to a level superior than someone starting their job fresh out of academia. According to Nissen and Levitt (2004), knowledge is considered an important resource in any modern enterprise. However, it is evenly distributed among the employees. The dynamic of knowledge is the ability to transfer certain knowledge through the enterprise between the organizations, the employees… Thus, emerges the idea of Knowledge Management (KM) in an attempt to describe different flows of different kinds of knowledge. Takeuchi and Nonaka (1986) have discussed multiple concepts of learning while working on projects, among which the most notable is the multifunctional learning where experts are encouraged to accumulate experience outside their own expertise. They gave an example of a printer that was developed by a team of mechanical engineers, which required them to learn electronics and design. Another example is a team of sales engineers that was required to design a Personal Computer; they were successful by going beyond their roles, learning from users and experts simultaneously.

Denkena et al. (2017) discussed in details the employees' competencies. The competencies are based on cognitive abilities, motivation, and practical skills. These factors are then weighed to calculate the suitability of an employee to a certain task. The employees are rated on a five-point scale for each task or operation. Learning effects are also taken into consideration through a step function which predicts competence development based on the number of executed operations. Denkena et al. (2017) based their learning formulas on basic functions developed by Wright (1936) and Levy (1965) who suggested the following Equation 4 (Wright Learning Formula) and Equation 5 (Wright Learning Formula - Limited).

$$t_n = t_1 \,.\, n^{-k} \qquad \text{Equation (4)}$$

$$t_n = c + \ (t_1 - c).\, e^{-k(n-1)} \qquad \text{Equation (5)}$$

Equation 4 & Equation 5 are applicable as a learning curve, where t is execution time according to the number of executions (n). k is the agent's learning rate. Equation 5 expands on this using c, a limit value. This literature's implementation of the learning process function was indeed useful for our paper; the formula suggested for the agent's learning is used and is on point considering how this paper views the learning process in an agile product development process. This will be discussed in detail in Chapter 3.

Starting with Nonaka (1994) and with Nissen and Levitt (2004) continuing his work, the latter built a flow that represents time and supports a multidimensional representation framework for analysis of knowledge-flow patterns. However, this attempt came up short as the modeling is the dynamics were static and not completely usable. Using Nonaka's (1994) Spiral model as a cornerstone, this section focuses on the model integration. Nonaka's (1994) model featured four enterprise processes. To quickly describe each, we can say that Socialization is experience sharing among team members; externalization is described as the use of metaphors to articulate knowledge and make it easier. Combination is the coordination between different groups in an organization (using documentation) to link and explain intra-team concepts. Internalization is different members in the organization combining their knowledge to produce work (mostly through learning while doing).

Second part of this model is the Nissen and Levitt (2004) Life Cycle Model. This model uses 6 phases to describe the knowledge flow: creation, organization, formalization, distribution, application, and evolution. Creation is new knowledge generation in an enterprise. Organization and grouping knowledge through repositories are the second phase. Third phase is transforming this knowledge to a formal representation. Fourth is sharing the earned knowledge through transfer. Fifth is the effective knowledge use for solving and advancing. Finally, in the sixth phase, is the evolution and refinement of knowledge. Based on Nissen and Levitt (2004) and Nonaka (1994), we can state that their attempt to dive into knowledge dynamics and knowledge flow is represented in two sections of this paper. While the knowledge transfer and sharing is interpreted in the Coordination section of this paper, the creation and application phases are useful for the learning process and the agent learning modules implemented in Chapter 3.

2.2 Simulation Techniques

In this section, we look into different simulation techniques that could be useful for our model. The first simulation environment to consider is System Dynamics (SD). It is an approach to problem solving developed in the early 1960s. A system is defined as a collection of continuously interacting elements whereas the dynamics is the change over time. In SD, the components or the elements are called the structure of the system. This structure defines performance and explicitly shows the relations among the elements. By doing so, any change in one element could affect multiple others. Another concept in SD is the Mental Model, which indicates that every person that interacts with a certain process would have its own model of this process. By sharing this mental model among them, the people would be able to improve the system or the process by simply reaching consensus. As for its validity, SD models could be considered as a best guess among the participants based on their understanding of the system (Sweetser - 1999).

Second simulation approach to consider is the Discrete Event Simulation (DES). DES is mainly implemented in systems where changes of states or values happen in interest points at discrete points in time, rather than continuously. (Fishman, 2013). To better explain this, an example of a bus picking up passengers is considered.

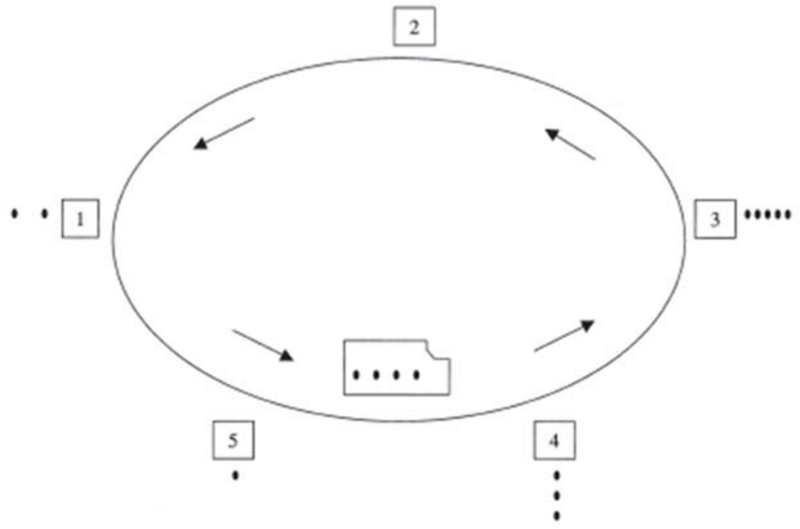


Figure 0-4 Discrete Event Simulation – Bus Example

Based on Figure 2-4, let's assume we have five bus stations with passengers waiting, the changing factors are the following: Location of the bus, number of passengers on the bus, number of passengers waiting at each station. At every station, when the bus arrives, we could assume that the number of passengers on the bus will change, as well as the number of passengers waiting at the station. This example describes the DES and emphasizes the discrete modeling as a non-continuous way of thinking since changes will only happen at certain points. (Fishman, 2013).

Another modeling technique is the Monte Carlo (MC) Simulation. Monte Carlo is a widely used simulation technique since it is able to handle variant needs. It mainly focuses on using mathematical algorithms to simulate and study quantitative analysis and decision making. It allows the users to see multiple possible outcomes of their models; using probability distributions, the system will calculate the results over and over with using different random values for each calculation to finally produce a distribution of possible outcomes. Even though it could be used a valid simulation model, Monte Carlo focuses more on the probability of risk related system as whole rather than what is needed – focus on the individual team member and its effect. Monte Carlo models assume that the process behaves stochastically based on certain logic specific to the model (Maier et al., 2014).

Agent-Based Modeling (ABM) is a modeling framework for simulating dynamic processes with autonomous agents. The agent makes its own decisions without any external interference (Klügl and Bazzan, 2012). This type of simulation is mainly used for decision making situations – in our case the choosing of a task to perform. Klügl and Bazzan (2012) considered that the Agent-Based Model is built on three elements: Agents, Relationships, and Environment. Agents are simply the subjects of the simulation. Their actions and behaviors are studied in the simulation and usually have the following attributes, regardless of the simulation.

'Autonomy' means that the agents are completely independent and self-driven; their decisions are automatic and simulated. 'Modularity' means that an agent is usually unique and different from the other agents. 'Sociality' means that an agent has interactions with other agents. And finally, 'conditionality' means that an agent would have a state determining its attributes and behaviors. Relationships determine how the agents are related and interact with one another. Interactions between agents will determine the simulation as the environment will be made by agents of different types and all results depend on how agents deal with one another. Garcia (2005) talked about the different uses of ABM in Innovation and focuses on the benefits and advantages of using ABM while also mentioning the suitable environment for ABM. Garcia (2005) compared ABM to other simulation techniques and pointed out that the ease of use for ABM represents a massive advantage. This ease of use is due to the lack of complex equations and statistics, primarily because ABM focuses on individual agents; noting that as rules are added on the system or on the agent the system complexity would increase slowly. Al Hattab and Hamzeh (2016) considered ABM as a method to analyze the collective behavior of agents within a system by allowing the agents to take individual decisions. Based on that, they created an ABM where the agents were people with a set of interchanging states which change depending on interactions among the agents. Also, while using ABM, Awwad et al. (2014) benefited from having heterogenous agents with different capabilities and goals. They stated that ABM offers the ability to test various scenarios with minimal effort.

Zankoul et al. (2015) studied the difference between DES and ABM in a construction site, considering the logistics and resources used. They modeled the DES as a sequence of discrete events where the entities go through the process sequentially. They also built an ABM simulation consisting of four agents with their own states which could affect the other agents. For example, the truck (Agent A) is to be filled using the excavator (Agent B). Even though the results of their models were almost identical, Zankoul et al. (2015) stated that using ABM instead of DES presented advantages such being easier to understand, more flexible, as well as having a better model performance.

Having seen these three simulation techniques, Agent Based was selected as it would indeed focus on team members and setting them as agents and considering the entire process as a flow of interactions and actions taken by the agents.

2.3 Existing Simulation Models for Product Development Processes

In this section, we are going to look into some existing simulation models that handle PD processes with a focus on task allocation. In the specific case of Software Product Development, Ngo-The and Ruhe (2009) explained the process of assigning tasks to developers. With the vast differences in skills among the software developers, they introduced an average skill level with a factor of 1.0 for each type of task. With a managerial judgment, the developers will get a productivity factor prod (d, q) where d is the developer and q is their ability to perform the task – and if yes how productively will they perform it. They propose a productivity vector ‘u’ which uses 4 factors: the developer d, time t, task n, and productivity factor q. This results in the following definition of u: $u(d, t, n, q)$ .This vector will be set to 1 if at a time ‘t’ the developer ‘d’ is working on task(‘n’,’q’). The task factor depends on the task ‘n’, and on the productivity factor ‘q’.

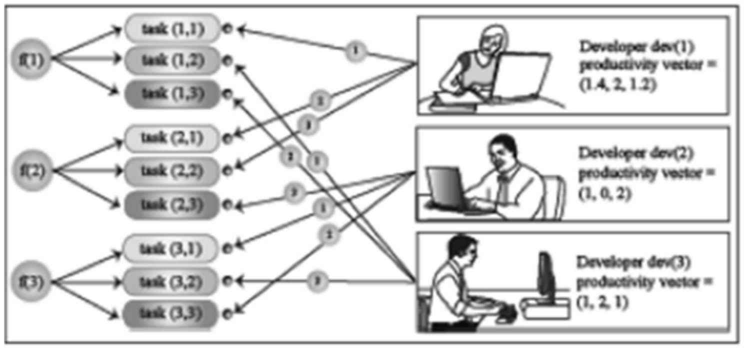
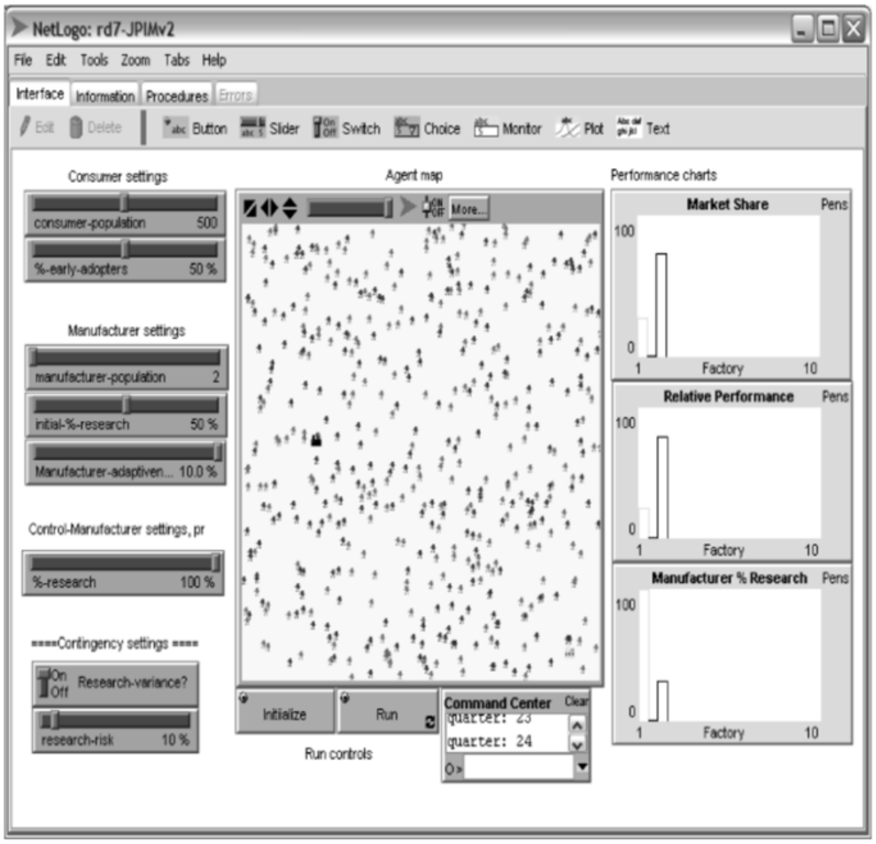

f(1)
f(2)
f(3)
task (1,1)
task (1,2)
task (1,3)
task (2,1)
task (2,2)
task (2,3)
task (3,1)
task (3,2)
task (3,3)
Developer dev(1) productivity vector = (1.4, 2, 1.2)
Developer dev(2) productivity vector = (1, 0, 2)
Developer dev(3) productivity vector = (1, 2, 1)

Figure 0-5 Task Distribution - Ngo-The and Ruhe (2009)

The human resource assignment is then done following Figure 2-5 - which represents an example of three developers to be assigned to nine different tasks for three different features, using the productivity vector. The best possible assignment considers the productivity of the developers, their availability, and the possible dependencies between tasks. This literature's suggested solution to task allocation is very accurate, since it uses a productivity vector which looks into the developer's time and productivity factor. However, our paper will go a bit further with the cooperation feature which allows multiple developers (agents) to work on the same task simultaneously.

Garcia (2005) also used ABM to develop a model that would study how manufacturers compete for customers using "Innovative Products" and "Incremental Products". Having two types of agents - late adopters and early adopters, Garcia built a Netlogo model that studies development, research, performance, and resource allocation.



consumer-population 500
%-early-adopters 50 %
manufacturer-population 2
initial-%-research 50 %
Manufacturer-adaptiven... 10.0 %
%-research 100 %
Research-variance? On Off
research-risk 10 %

```
quarter: 23
quarter: 24
```

Figure 0-6 Netlogo Model - Garcia (2005)

The Netlogo model in Figure 2-6 features the input settings (Consumer, manufacturer, control-manufacturer, and contingency), the agent map where the agents move and behave based on their rules, as well as the performance charts. In the model's rules specifications, Garcia (2005) set the agent's movement and engagement rules which are based on theoretical studies and assumptions, as well as case studies. Also, to be defined were the manufacturers' rules; basically, the manufacturer's strategy depended on the consumer demand while also considering the limited resources available. A trade-off was in order between the two different types of projects based on the consumer demands.

2.4 Agile and Scrum Project Management

Agile project management has been around for a while now and is becoming a popular management strategy for various types of PD processes. It is originally a software project management process -introduced in 2001 through the Agile Manifesto- that focuses on fast iterations as a way to break down large projects into releasable iterations. This facilitates complex projects by dividing the complexity into simpler and easier iteration, which makes the development process easier. Another advantage is that by applying iterations and frequent feedbacks, the product in development is being regularly reviewed and it would make it easier to keep it on track. Also, it has a high adaptability rate. The iterative approach allows modifications or incrementations in each iteration. (Hughes, 2019)

Even though it is considered to be limited to software PD, it could be useful for any PD process. Cooper and Sommer (2016) discussed this point and believed that the agile methods can be integrated with gating approaches to adapt to a product manufacturing process. They introduced the hybrid model of Agile-Stage-Gate to benefit from the beforementioned agile advantages, as well as the existing Stage-Gate which is designed to assist in selecting the right projects within a company and decide key stages and roles. As a first step, they studied the compatibility of Agile and Stage-Gate and found that the Agile methods complete the stage-gate model by providing day-to-day planning and progress reporting which it lacks. Their second step was to see if this hybrid model could work for physical products. They based their findings on Sommer et al. (2015) which researched five manufacturing firms that implemented the hybrid model and found that the firms have had improved efficiencies and reduced work and rework efforts.

Sliger (2011) defined scrum as an agile method of iterative and incremental product delivery method that relies on regular feedback and collaborative decision making. She also went on to explain the scrum framework. The process begins with a product backlog; a set of features required by the client/customer referred to as the Product Owner. The next step is the sprint planning meeting, but before going into that, let us define the sprint. It is basically a defined time window – usually between one and four weeks- in which work is done on the backlog's features. The sprint planning meeting happens before launching the sprint and it is in this meeting that the team decides which features are to be worked on during the sprint. This leads to the creation of a sprint backlog. Following the meeting, the sprint is launched and work on the tasks begin. During the sprint, the team attends a short daily meeting (usually around ten or fifteen minutes long) known as the daily scrum in which the members talk about what they did the previous day, and what they plan on doing today and what issues they are facing. At the end of the sprint, the team displays their work to the Product Owner and receives feedback to adjust the next sprint. The team also has a retrospective meeting in which they discuss the sprint and how to improve for the following sprint. Figure 2-7 shows the multiple stages of a Scrum product development process discussed in this section.

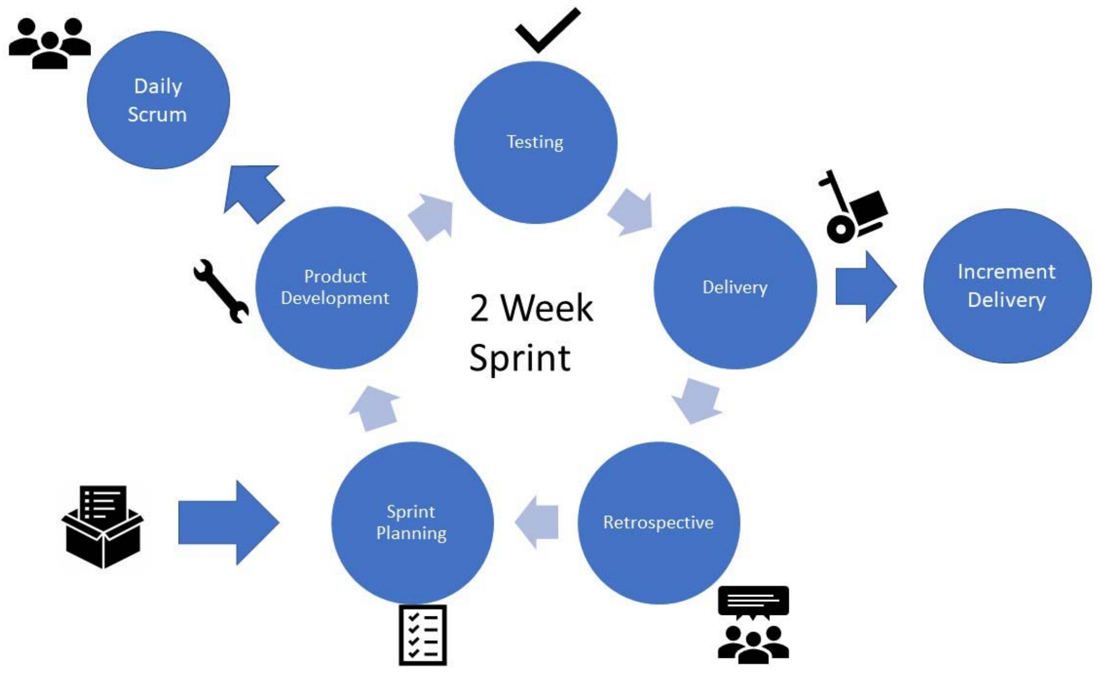


Figure 0-7 Scrum Product Development

In our study, Figure 2-7 could be represented as follows: an input file to the model containing all the tasks selected during the sprint planning phase, as well as their attributes. The product development phase and the testing phase are part of the tasks to be completed during the simulation. The delivery phase would simply be the end of the sprint. The retrospective and the daily scrum are represented by the meetings performed by the agents and the client.

CHAPTER 3

AGENT-BASED MODEL FOR RESOURCE ALLOCATION & INTERACTION IN PD

Consider a PD project that has I $(1 \leq i \leq I)$ participants and J $(1 \leq j \leq J)$ design activities or tasks. These I participants shall be referred to as agents. As beforementioned, agent-based modeling is advantageous in our context since our focus is on the individual agent (Garcia, 2005). Agent-Based modeling first requires defining the I agents; by initially specifying them. They are the drivers/decision-makers of the system. Next, is to define their behaviors. The behaviors of the agents will depend on the system's settings and how each agent should respond to their respective situations in the simulation model. Third, is to determine the relationships among the agents and to add the methods controlling their interactions. The team members will be the human employees that will go from one task to another to accomplish the tasks. They will each have specific attributes that determine their efficiency with values ranging from 1 to 10 (their technical skills, their managerial skills, and their overall experience), their learning ability l – which depends on each agent's character, and their available time to work. The learning ability l ranges from 0 to 10. The value 0 would indicate an agent with no ability to learn, 5 would indicate an average learner and 10 is for a very fast learner. The agent's available time Z_i indicates the time during which the agent is available during the project.

The tasks, on the other hand, will be waiting for employees to perform them. Each task also has a set of skills ratings, which are the needed skills and their minimum level. Additionally, each task has the minimum, medium, and maximum estimated time for completion and its priority factor. Furthermore, the tasks have weights for the two skills – technical and managerial - as well as a weight for the overall experience attribute and another for the priority level used only for the dependency priority. These weights indicate how important each factor or skill is for the selection of the task. By using a scale of 0 to 1, where 0 is lowest and 1 is highest, a weight of 0.9 out of 1 will emphasize the importance of this factor for the task. These weights and how they affect the task allocation method will be explained more in the following section. Also, the tasks have a rework probability, a rework impact, dependency relations and their respective dependency strengths and impacts – all of which will be explained in the next sections.

To discuss the tasks' status, any given task j could have one of the following statuses. Pending tasks are the main type of tasks which are available or ready to be worked on. The dependent tasks require prerequisite tasks to be completed before they become available for working on them. The completed tasks are simply the tasks that one or more agents have worked on and completed. Finally, the rework status for the tasks is assigned to tasks that have been completed, but due to changes or new requirements, have been reset to be repeated. Having explained the setup, the upcoming sections will explain the agent's behaviors and simulation flow.

3.1 **Task Allocation Method**

The primary challenge in developing the task allocation method is determining for each agent i which task from the full set of tasks J, is the suitable task to be performed. To address this, we propose Equation 6, called the "skill difference coefficient" (SDC) denoted by $\Delta_{ijn_iq_i}$.

$$\Delta_{ijn_iq_i} = w_j^m \left|A_{in_iq_i}^m - T_j^m\right| + w_j^t \left|A_{in_iq_i}^t - T_j^t\right| + w_j^l \left|A_{in_iq_i}^l - T_j^l\right| \quad \text{Equation (6)}$$

This equation is based on the formula suggested by Stobrawa et al. (2018) discussed in Section 2.1.2. However, it has been adjusted according to our needs. $\Delta_{ijn_iq_i}$ represents the skill difference coefficient for agent i and task j while considering the number of tasks completed n_i by agent i - since the talents of agent i would change depending on the number of tasks completed and rely on the last task completed by agent i, q_i. All the parameters in Equation 6 and their possible values are represented in Table 3-1. (A table containing all the variables is available in the Appendix A)

Parameter	**Definition**	**Values**
$\boldsymbol{i}$	Agent i	$1 \leq i \leq I$
$\boldsymbol{j}$	Task j	$1 \leq j \leq J$
$\boldsymbol{n_i}$	Number of tasks completed by agent i	$i \geq 0$
$\boldsymbol{q_i}$	Last task completed by agent i	$1 \leq q \leq J$
$\boldsymbol{\Delta_{ijn_iq_i}}$	Skill Difference Coefficient between agent i and task j considering n_i and q_i	$0 \leq \Delta_{ijn_iq_i} \leq 10$
$\boldsymbol{w_j^m}$	Weight of managerial knowledge on task m	0 – 1
$\boldsymbol{w_j^t}$	Weight of technical knowledge on task j	0 – 1
$\boldsymbol{w_j^l}$	Weight of overall experience on task j	0 – 1
$\boldsymbol{A_{in_iq_i}^m}$	Managerial talent for agent i for n_i completed tasks, with q_i being the last task completed by agent i	1– 10
$\boldsymbol{A_{in_iq_i}^t}$	Technical talent for agent i for n_i completed tasks, with q_i being the last task completed by agent i	1 – 10
$\boldsymbol{A_{in_iq_i}^l}$	Overall Experience for agent i for n_i completed tasks, with q_i being the last task completed by agent i	1 – 10
$\boldsymbol{T_j^m}$	Required managerial skill on task j	1– 10

$\boldsymbol{T_j^t}$	Required technical skill on task j	1 – 10
$\boldsymbol{T_j^l}$	Required overall experience on task j	1 – 10
$\boldsymbol{Z_i}$	Available table for agent i	$Z_i \geq 0$

Table 3-1 SDC Attributes

To explain Equation 6, consider a situation where we are trying to choose the most suitable task for agent i among the J tasks. The weights vary between 0 and 1 ($0 \leq w_j^m, w_j^t, w_j^l \leq 1$) with 0 being the lowest weight and 1 being the highest, with the sum of all three weights being equal to 1 ($w_j^m + w_j^t + w_j^l = 1$). These weights represent the importance of the specified skill to this task – how much this skill weighs for the entire task; they should be set by the project managers that are assessing the tasks for their allocation. Their expertise should allow them to define the values for the weights accordingly.

The talents and skills are scaled from 1 to 10, to represent the different levels of skills. 1 being the lowest grade and representing the novice, 10 being the highest – representing the expert, 5 being the average grade representing a mediocre person, and the grades in between represent different shades of expertise. Using these weights for each attribute of task j (i.e., managerial w^m, technical w^t, and overall experience w^l), the formula will determine the difference between the talents $A_{in_iq_i}^m, A_{in_iq_i}^t, A_{in_iq_i}^l$(managerial, technical, overall experience) with respect to the number of tasks completed by agent i (n_i) and the last task completed by i (q_i), and the minimum required skills for task j (T_j^m, T_j^t, T_j^l), then it will multiply each difference ($A_{in_iq_i}^m - T_j^m, \ldots$) with its respective weight.

Additionally, let us consider the case where there exists a priority effect. For each prerequisite task j, there exists have a priority weight w_j^p($0 \leq w_j^p \leq 1$). Additionally, P is the priority factor applicable only for tasks that have dependent tasks ($0 \leq P \leq 10$). P indicates the importance of finishing the prerequisite tasks in order to convert the dependent tasks into an available status. To focus on the importance of having a priority, we introduce the selection coefficient $\mu_{ijn_iq_i}$.

$$\mu_{ijn_iq_i} = \begin{cases} \Delta_{ijn_iq_i} - w_j^p P \; for \; prerequisite \; tasks \\ \Delta_{ijn_iq} \; otherwise \end{cases} \qquad \text{Equation (7)}$$

Equation 7 represents the selection coefficient $\mu_{ijn_iq_i}$ which combines the skill difference coefficient $\Delta_{ijn_iq_i}$ with the additional priority factor P and the task's priority weight w_j^p. Including the priority factor would lower the value of $\mu_{ijn_iq_i}$ and favor the task j for selection. This factor only affects prerequisite tasks based on their priority weight w_j^p. In case a task is not a prerequisite task, the value of w_j^p would be 0. By applying the priority factor, we are decreasing the value of $\mu_{ijn_iq_i}$. Since our aim is to get the closest match between the agent's talents and the task's skills, we will rely on the results of Equation 7. It will allow us to select the task j where $\mu_{ijn_iq_i}$is the smallest as the chosen task to work on by agent i. In this sense, the closer the agent's talent value is to its respective task's skill value, the smaller value of $\mu_{ijn_iq_i}$ is obtained. To specify the selection of tasks to agents, $\mu_{ijn_iq_i}$ is calculated for each agent i task j combination. As an example, consider the following selection coefficient matrix φ,shown in Figure 3-1, for two agents and two tasks.

$$\varphi = \begin{bmatrix} & \boldsymbol{j_1} & \boldsymbol{j_2} \\ \boldsymbol{i_1} & -2 & -1.5 \\ \boldsymbol{i_2} & -3 & -2 \end{bmatrix}$$

Figure 0-1 Hypothetical Selection Matrix

Considering the matrix in Figure 3-1, the selection would be done starting by choosing the smallest coefficient in the matrix and allocating that agent to the task. Then the column of the task and the row of the agent are removed from the matrix – this is repeated until all agents have been allocated tasks. Starting with $\mu_{2\,1\,n_1q_0}$ of agent 2 and task 1, we allocate task 1 to agent 2 since it has the smallest value of the selection coefficient in the matrix. Next, we consider task 2 ($j = 2$). Even though agent 2 ($i = 2$) has the smallest μ for task 2, agent 2 would be assigned to task 1 ($j = 1$) since their selection score is even less. In this case, task 2 would be assigned to agent 1.

With a task j determined for agent i, we need to find the difference between its talents and the task's required skills in order to estimate its performance on the task. Therefore, we introduce the task execution coefficient $\omega_{ijn_iq_i}$ in Equation 8.

$$\omega_{ijn_iq_i} = w_j^m\left(A_{in_iq_i}^m - T_j^m\right) + w_j^t\left(A_{in_iq_i}^t - T_j^t\right) + w_j^l\left(A_{in_iq_i}^l - T_j^l\right) \qquad \text{Equation (8)}$$

Equation 8 represents the task execution coefficient $\omega_{ijn_iq_i}$ which is used to determine how the agent would perform on their task. Without the priority factor, and by using the actual difference (positive and negative), the $\omega_{ijn_iq_i}$ focuses only on the difference between the agent's talents and the task's required skills at a certain number of completed tasks n_i. This allows us to reflect on the actual difference between the agent i and the task j, and present a real task execution simulation. By having a positive value of $\omega_{ijn_iq_i}$ we could determine that agent i would perform the task better than average. Similarly, a negative value of $\omega_{ijn_iq_i}$ would mean that agent i would perform below average.

3.2 Meetings & Collaboration

This feature is designed to simulate the effect and behavior of having a team meeting, from a time consumption perspective and a performance perspective. As previously mentioned, Dingsoyr et al. (2018) talked about the importance of physical meetings and how they enrich communication and collaboration. According to Moe et al. (2018) – at least for the Agile process - we should include and take into consideration the daily scrums and the iteration planning meetings. This will be reflected in our PD model by having two types of meetings, one that covers collaboration and communication, and another that covers client meetings.

For the first type, we will consider a daily meeting among agents, called a daily scrum in Agile process. During this meeting the agents would talk about what they have accomplished or worked on during the previous day and discuss difficulties they are facing. Collaboration could happen among the agents in such a meeting as a form of knowledge and information exchange. The collaboration could simply happen by having an agent suggest a solution to another agent's problems which would make their task a bit easier and increase the agent's $\omega_{ijn_iq_i}$. The effect of this collaboration will be reflected in a random factor m called meeting factor; this factor could vary between positive and negative. ($-1 \leq m \leq 2$, the set of values of m is explained in Section 3.6.4). Having a positive factor would indicate that the collaboration had a positive outcome on the task completion, and the agent's task execution coefficient $\omega_{ijn_iq_i}$ increased by m. A negative factor would mean that some lack was discovered due to the collaboration thus the agent's $\omega_{ijn_iq_i}$ decreased by m. The frequency of such a meeting is denoted as α_{me}.

The second type of meetings would be the client meeting. This type of feature would simulate having meetings with clients, an example would be the retrospective in the Agile process which discusses the tasks completed within the last sprint. As experienced in real-life circumstances, this meeting has a chance of requiring task rework; this is basically a request by the client for a change in an already completed task and thus returning its state to pending or in need of work. The probability of a rework happening, and the percentage of rework required in case of a rework will both be values inputted by the user in the simulation model. The first percentage (rework happening) is ρ_{j,k_j} and represents the probability of a client requiring changes after the completion of a task j for the k^{th} iteration of j, and the second is called δ_j (Impact of task rework) and represents the percentage of rework required due to the requested changes on the same task j. The rework probability ρ_{j,k_j} is set by the project manager to estimate the possibility of having to rework on task j. It is also related to the client meeting frequency. The project manager's estimate should take into consideration the client meeting frequency α_{mc}. Having more frequent meetings would lower the rework probability since the client would have more information and could provide early feedback on possible changes and reworks. The frequency of the client meeting is denoted as α_{mc}.

During both types of meetings, we will assume that within the agents' communication amongst themselves, as well as their communication with the client, some type of negotiations take place. During these negotiations, the participants would discuss their solutions or ideas in order to find a common ground – through compromise or persuasion. To take this into account, the meeting time (also considered as Meeting Cost) will be reduced from each agent's remaining time. The meeting costs for team meetings and client meetings are denoted β_t and β_c, respectively.

3.3 Cooperation

Cooperation is an important feature discussed in this paper. As defined in the introduction, it is the process of allowing more than one agent to perform a single task. This feature represents the possibility of having a difficult task, that requires a lot of time for one agent to complete, being completed by two or more agents. Let us first define what will be considered as a difficult task in our simulation. When determining the task to be performed by the agent, we defined a threshold called the cooperation threshold h. It is a user input value ranging from -5 to 0. It will be compared with the agent's $\omega_{ijn_iq_i}$ (since it will focus on only talent versus skills) - for each task – to decide whether the task should be available for cooperation or not. Simply put, if $\omega_{ijn_iq_i} <$

h, the task would be available for other agents to work on. For example, having an agent i and a task j with an $\omega_{ijn_iq_i} = -2$ (-2 being an arbitrary value of the $\omega_{ijn_iq_i}$), comparing this $\omega_{ijn_iq_i}$ to h (consider $h = -3$) might indicate that task j is not worth having cooperation – more than one agent working on it. In case the task required cooperation, the same algorithm that chooses the task allocations will be applied but it will take into consideration the already assigned tasks that need cooperation. An extra feature available is to restrict having more than Y agents on the same task. By determining $Y,$ we are limiting the maximum number of agents working on the same task. In our model, we are setting $Y = 2$ when there exists cooperation. This variable could change based on the project or the company. In case there is no cooperation allowed, the value of Y is automatically set to 1.

3.4 Agent Learning

In this section, we consider how improving agents' talents will affect task execution and the PD process. To do this, we propose a learning formula to estimate the value of learning from each task completed. The execution time for each task is multiplied by (n^{-k}) to reduce the time, where n in the number of tasks completed and k is the agent's learning ability (Denkena et al. - 2017). We build on this relationship to reflect how the number of executions will advance the agent's talent (thus lowering the execution time of any related task). We assume that agent i improves whenever it finishes a task j. To determine its improvement on each talent, we use the weight for each skill. Since it represents the importance of the skill for the task, we consider that by finishing this task j, agent i will have improved its skills ($A^m_{in_iq_i}, A^t_{in_iq_i}, A^l_{in_iq_i}$) in a relative manner to the task's weights (w^m_j, w^t_j, w^l_j). To do this, we will use Equation 9 that will affect each of the agent's talents (respectively to their weights) for each time the agent finishes a task and improves its talents.

$$A^{s}_{i,n_i+1,q_i} = A^{s}_{in_iq_i} + \frac{w^{s}_{q}}{\left(n_i^{\frac{1}{l_i}}\right)} \qquad \text{Equation (9)}$$

In Equation 9, A^{s}_{i,n_iq_i} is the agent's talent of type s(where s is an index for the specified skill; s = {m, t, l}) for the current number of tasks completed n_i by agent i, considering that the completed task is q_i (the current agent's task). w^{s}_{q} is the task's skill weight, and l_i is the agent's learning ability –defined in Chapter 3. Equation 9 allows the agent to improve their talents based on the type of task it just completed. A task q with a high weight in technical skill would be of technical nature, and by completing it, agent i would improve its technical talent the most, based on the task's technical weight w^{t}_{q}. The same applies for a task with a focus on managerial or a higher experience level. All the agent's talents will improve relatively to the weights on the task.

3.5 Task Dependency

Dependency among the tasks is an important factor in any PD process. It indicates the development flow which requires a certain sequence of execution. As discussed in Chapter 2, task dependency is a feature in this study and allows the realization of a specific flow in the PD process. Since the tasks' details are imported for each simulation (as explained in Section 3.6.1), the dependency relation, strength and impact between the tasks are based on the imported data. As a simple example, suppose task j depends on task v. If so, task j will not be available to be performed or worked on until the completion of task v. We define the dependency strength τ_{jv} and the dependency impact ε_{jv}. In case there exists a dependency relation between task v and task j, τ_{jv} is the strength of this dependency; in other words, the probability of having to work on task j when rework is performed on v. And in case there exists rework on task j due to the rework on v, the dependency impact ε_{jv} represents the percentage of rework needed (i.e. what fraction of task j is repeated). These attributes are mainly used for the rework feature and their importance is explained in Section 3.6.4.

Additionally, the same parameters are used to implement the feedback rework among tasks. Suppose that upon the completion of a dependent task j, we also implemented changes involving the prerequisite task v. In this case, the dependency strength and impact between tasks j and v are used to determine the possibility and amount of rework on task v. The details are explained in Section 3.6.4.

As for the effect of dependency in the task allocation process described in Section 3.1, we introduce a dependency priority P. This factor is a user input value ($0 \leq P \leq 10$), which determines the priority level of the pre-requisite tasks such as v. The higher it is, the more importance is given to the prerequisite tasks. Its value is between 0 and 10, 0 indicating that the dependency has no effect on the allocation process, and a max value of 10 to represent urgent prerequisite tasks. Therefore, the value is subtracted in the selection coefficient formula, as means to prioritize task v ahead of another task which could have a better selection coefficient but is not a prerequisite to any other task.

3.6 Implementation

This section explains how the model works using the Agent-Based Simulation software Netlogo. It will cover the creation of agents and tasks, the task allocation method, meetings, task dependency, as well as agent learning. Diving into the details of the implementation will show how each of the beforementioned features is implemented and used, and how the model's options affect the model. Before discussing each feature, let us define the model settings as in Table 3-2.

Setting Name	Setting Value	Description	Parameter
Agent Learning	ON/OFF	Agents improve their skills	-
Cooperation	ON/OFF	Allowing agents to work together	-
Cooperation Agent Limit	$Y \geq 1$	Maximum number of agents allowed to work on the same task	Y
Cooperation Threshold	$-5 \leq h \leq 0$	If > ω the task is eligible for cooperation	h
Dependency	ON/OFF	Dependency between the tasks	-
Dependency Priority	0->5	Value of Dependency priority in μ	P
Meetings Allowed	ON/OFF	Simulating a team meeting	-
Meetings Frequency	$0 \leq \alpha_{me} \leq 50$	A team meeting happens every α_{me} time steps	α_{me}
Meetings Duration	$1 \leq \beta_t \leq 10$	Duration of team meeting (in time steps)	β_t
Client Meetings Allowed	ON/OFF	Simulating a Client meeting and their feedback	-
Client Meetings Frequency	$0 \leq \alpha_{mc} \leq 100$	A team meeting happens every α_{mc} time steps	α_{mc}
Client Meetings Duration	$1 \leq \beta_c \leq 10$	Duration of client meeting (in time steps)	β_c

Table 3-2. Netlogo Model Settings

3.6.1 Agent and tasks creation

The first step of the model is to setup the agents and tasks. The details of each user and each task are imported from comma separate values (CSV) files as soon as the user clicks on setup (Available in Appendix B). The system will create the model by setting the agents' names and initial coordinates as well as setting their color to green to indicate their status as ready and available. The agents will then be created with different availability times, skills, and learning ability; based on the specifications determined in the file.

Similar to the agent creation, the tasks are created based on a second user imported CSV file. Based on their details, tasks without a prerequisite will be colored in grey to indicate that they are pending and ready to be performed. If the task dependency feature is enabled and the imported task has a prerequisite based on the dependency relation θ_{jv}, it is considered as a dependent task and will be colored in blue. Note that every dependency has its own object/record to allow for multi-dependency among the tasks.

3.6.2 Task Allocation

As explained in Section 3.1, the task allocation method requires calculating - for each agent - the difference between the agent's acquired talents and the tasks' required skills. This step happens automatically after setting up the agents and tasks and occurs on each time step for the agents that do not have an assigned task (explained in next section). Let us consider the initial task allocation after the creation step. For each agent i, the system will go through all the tasks in the set of tasks J. For each task j in the set J, if j is available by being colored in grey (status pending) or orange (status rework – discussed in Section 3.6.4), $\mu_{ijn_iq_i}$ is calculated and if the calculated $\mu_{ijn_iq_i}$ is the lowest among the tasks, j is set as the agent's task. As explained in Section 3.1, the selection process allocates the tasks to the agents based on their lowest combination scores, starting with the smallest values of $\mu_{ijn_iq_i}$. Technically, along with assigning the task, its coordinates are used to place agent i next to it. This is specifically done in Netlogo as to have the task j adjacent to the agent i. Being in a two-dimensional simulation, j's coordinates are along the x-y axis. Having i and j in adjacent positions is used to show each agent's assigned task at time r.

Also taken into consideration is the cooperation feature which allows multiple agents to be allocated to a single task. If the cooperation is allowed, $\mu_{ijn_iq_i}$ is compared to the cooperation threshold h, to indicate if j needs can solicit the help of other agents. That way, when the system is choosing a task for another agent i', task j is checked for allocation even though it has an assignee. An additional option is to restrict having more than two agents performing the task. By enabling this feature, each task j could at most be assigned to two agents.

3.6.3 Simulation Process

The system's time steps can be considered as any unit of time. Let us define time r as the value of the system clock – note that r will be considered in hours. At each time step, the system checks if all the tasks are completed or all agents are out of time, if any of these conditions is true, the system will stop and the simulation ends. Once the simulation stops, the system will indicate whether the simulation was successful or not. The simulation is considered successful if all the tasks are completed at the end of the simulation. If the simulation stopped because the agents were out of time, the simulation is considered not successful. Otherwise, the system will check r to see if it is time for a team meeting or a client meeting. Referring back to Section 3.2, the system will check if the time value r is a multiple of any of the meeting frequency values (team meeting frequency α_{me}, or client meeting frequency α_{mc}). If so, the system will launch one of the meetings processes, which will be discussed thoroughly in Section 3.6.4. If the time r does not match any of the previous conditions, the system will launch the process of task execution and task allocation.

The task execution process goes through all the agents iteratively and checks that each agent i still has time available; otherwise, the agent will be colored in red and will stay still as i is then considered to have used up all its available time. Another condition is to check if agent i has an assigned task, if not, i will be colored in blue and stay still as it waits for a task to be assigned to it. To focus on executing tasks, let us consider that agent i has time remaining and has an assigned task j. Our model assumes that there exists some time spent by agent i to select task j (as a setup time), which can be considered idle time by agent i (i.e. wasted). The agent's available time is reduced by a factor χ – the idle time – for each task selection. This factor is explained in Section 3.6.5.

To consider the task execution process, we will note the remaining time for task j as $t_{jrn_ik_jq_i}$. If $t_{jrn_ik_jq_i} > 0$, j still has work to be performed, and the agent's previously calculated Skill Difference Coefficient $\omega_{ijn_iq_i}$ is applied to find how much the remaining task time $t_{jrn_ik_jq_i}$ should decrease for every time step. In this sense, the remaining task time $t_{jrn_ik_jq_i}$ would be reduced by a value relative to $\omega_{ijn_iq_i}$. Considering that for an the execution coefficient $\omega_{ijn_iq_i} = 0$, $t_{jrn_ik_jq_i}$ should be reduced by 1 for every time step since this would be the average value of matching between the agents' talents and the task's skills. Furthermore, taking the extreme values of $\omega_{ijn_iq_i}$ (ranging from -10 to 10) and considering that for $\omega_{ijn_iq_i} = 10$ agent i should be reducing the task time $t_{jrn_ik_jq_i}$ by 2 (twice as fast as an average performer), we propose the following Equation 10 to calculate the task remaining time $t_{jrn_ik_jq_i}$ based on $\omega_{ijn_iq_i}$.

$$t_{j,r+1,n_i,k_j,q_i} = t_{jrn_ik_jq_i} - (0.1\omega_{ijn_iq_i} + 1) \qquad \text{Equation (10)}$$

In Equation 10, $t_{j,r+1,n_i,k_j,q_i}$ is the remaining time for task j. Since $\omega_{ijn_iq_i}$ is related to the number of tasks completed (n_i) by agent i, the task time will also depend on n_i.Furthermore, the execution coefficient depends on the last task completed by agent i and thus the remaining time also depends on q_i. Additionally, the remaining task time will also depend on the task iteration k – explained in this Section 3.6.4. For $\omega_{ijn_iq_i} \geq 0$, $t_{jrn_ik_jq_i}$ would be reduced by a value between 1 and 2 (since the maximum value of $\omega_{ijn_iq_i}$ is 10). On the other hand, for a $\omega_{ijn_iq_i} < 0$, $t_{jrn_ik_jq_i}$ would be decreased by a value between 0 and 1. This would simulate the task execution process based on different talent values. An agent with talents less than the task's required levels would need more time to perform it, thus effectively reducing the remaining task time $t_{jrn_ik_jq_i}$ by less than 1 for every time step. The same applies for an agent whose talents are better than the task's required skills. The agent would be decreasing $t_{jrn_ik_jq_i}$ by more than 1.

Next, if $t_{jrn_ik_jq_i}$ =0, the task j is considered to be completed. j is then colored in green, and if the Agent Learning feature is enabled, i's talents will be increased as indicated in Section 3.4. A final check on the task level is the dependency. If j is a pre-requisite task to any other task v, v is then rendered available by being colored in grey (instead of blue) if all the pre-requisites of v are completed. Lastly, having completed the assigned task, i needs to determine the next task. The task allocation process is started again, and the flow continues.

3.6.4 Meetings & Rework

As mentioned in the previous section, meetings will occur based on certain time frequencies α_{me} and α_{mc}. The first case to consider is the team meeting. Suppose we are at a certain time in the process and it is time for a team meeting. To simulate a meeting between the team members, we would go through the agents and change their respective $\omega_{ijn_iq_i}$ randomly. The system will randomly select – with equal probability - between the following set of values: {-1,0,1,2}. These values are considered since they would have a moderate impact on $\omega_{ijn_iq_i}$ that would replicate an average effect of meeting communication. In other words, it has an equal probability of increasing the $\omega_{ijn_iq_i}$ by 2 or 1, decreasing it by 1, or leaving it as it is. As mentioned in Section 3.2, this would simulate the effect of communication between agents. Since communication in a meeting is more likely to help improve the tasks, we included two positive values ,one neutral value, and one negative value. The changes to the agents' $\omega_{ijn_iq_i}$ would only affect the task completion time (discussed in 3.6.3). By specifying this, communicating during the meeting could ease a task (by increasing the $\omega_{ijn_iq_i}$), could render it harder (by decreasing the ω_{ij,n_i}), or could have no effect on it. Additionally, the meeting duration β_t is decreased from the agents' respective remaining times.

The second case to consider is the client meeting, which occurs between the team members and the client. As mentioned in Section 3.2, it is during this meeting that rework is discussed. Suppose we are having a client meeting. The system will go through the completed tasks and based on their rework probability ρ_{j,k_j}(imported from the original task details file), the system will set the task j to be reworked. The new task time $t_{jrn_ik_jq_i}$ is calculated as indicated in Equation 11.

$$t_{jrn_ik_jq_i} = \delta_j * t_{j,0,0,0,0,0} \qquad \text{Equation (11)}$$

Noting that in this case $k > 0$ since this is a rework and it wouldn't be the first iteration of this task. δ_j is the rework percentage of j and $t_{j,0,0,0,0,0}$ is the original time of j. j would then need extra work to be performed and it would be available to be allocated, thus it would be colored in orange. Consequently, the rework probability ρ_j is reduced for every iteration. In this manner, we can apply Equation 12.

$$\rho_{j,k_j+1} = \rho_{j,k_j}/2 \qquad \text{Equation (12)}$$

Equation 12 signifies that for every iteration of rework, we would reduce the probability of having rework on the same task j by half. This represents the smaller chance of having to rework a task on which we have already had rework.

An extra feature to consider is the dependency in the rework. If v is a prerequisite task for j, when v requires some rework, it would be logical to assume that if j is completed, it could also need rework. With a probability of rework on j equivalent to the dependency strength

τ_{jv} (defined in Section 3.5), j would be set to available for work as well (also colored in orange), and its new task time $t_{jrn_ik_jq_i}$ would be set as follows in Equation 13:

$$t_{jrn_ik_jq_i} = \delta_v * t_{j,0,0,0,0,0} * \varepsilon_{jv} \qquad \text{Equation (13)}$$

In Equation 13, $t_{jrn_ik_jq_i}$ is the new task time of task j for its k^{th} iteration, δ_v is the same rework percentage of v, $t_{j,0,0,0}$ is the original task time of j, and finally ε_{jv} is the dependency impact between j and v. This would indicate the impact of rework on task j due to the rework on task v. Rework on j is related to the percentage of rework on v, as well as the dependency impact between them.

Another case of rework to consider is rework due to feedback. Consider the same scenario where task j depends on task v which in turn depends on task p. These relationships are illustrated in Figure 3-2.

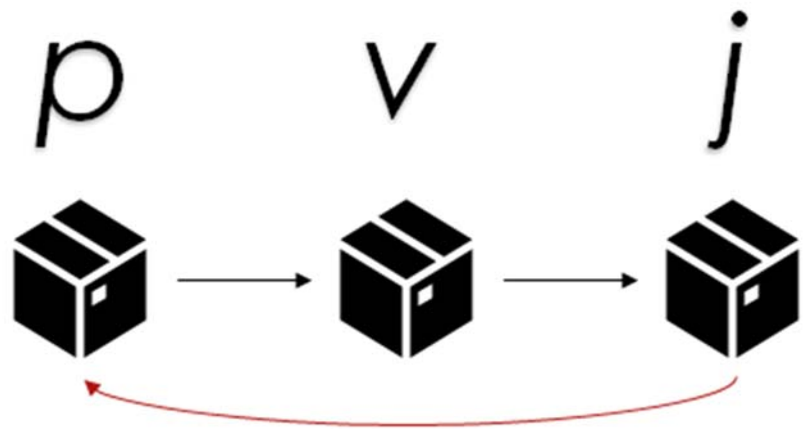


Figure 0-2 Rework from Feedback

Looking into Figure 3-2 and upon completion of task j, there were some modifications done that required a form of rework on task p – note that task p had already been completed since it is a prerequisite for v. The probability of this happening is equivalent to the dependency strength τ_{jp}. In case of rework required, task p would require additional time of work based on Equation 14.

$$t_{p,r,n_i,k_p q_i} = t_{p,0,0,0,0,0} * \varepsilon_{jp} * \delta_p \qquad \text{Equation (14)}$$

3.6.5 Idle Time

To simulate the time that agent is not actually working on a task, nor is it in a meeting, we introduce the idle time on the agent. Basically, it is the time to simulate breaks or time wasted during the day. The idle time for agent i will be incremented – and thus the remaining available time for agent i decremented, whenever agent i is selecting a task. That way, the agent's available time takes into consideration time spent without working. The idle time is a factor χ $(0 \leq \chi)$ where 0 indicates no time loss happens at all. In our model $\chi = 1$, but it could be a different value based on the company simulating the PD process.

3.6.6 Process Flowchart

Figures 3-3, 3-4 ,and 3-5 represent the entire process of the model (A complete version of the process flow is available in Appendix C). It starts by checking the tasks available for work as well as the agents available and that do not have any assigned tasks. The tasks available are the pending tasks, and the tasks that have been set to rework. The agents available are those who still have available time and are not assigned to any tasks. By calculating their skill difference coefficients and their selection coefficients, the selection matrix is built and then the task is assigned to the most suitable agent. With the agent's task determined, the execution coefficient is calculated.

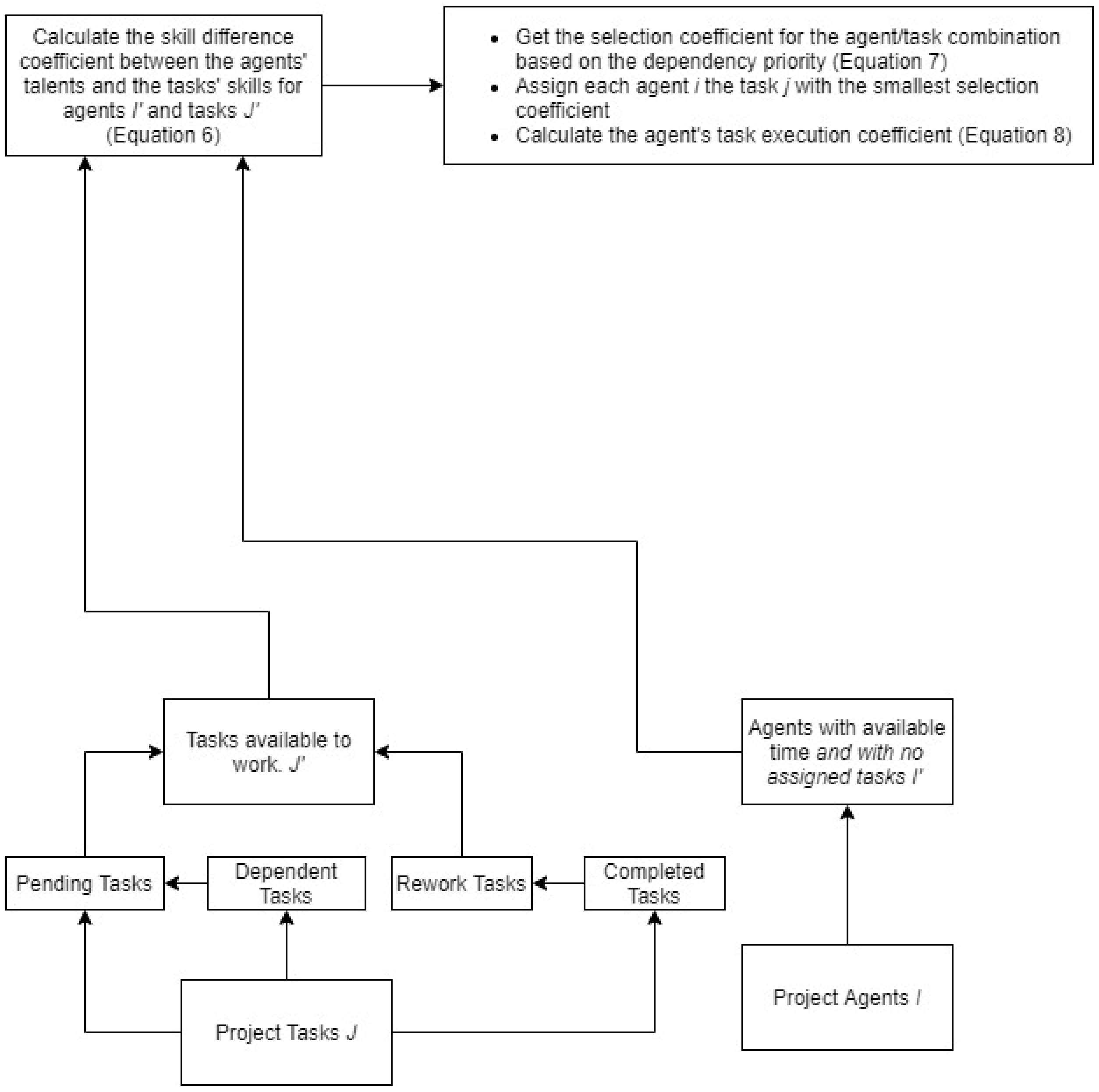

Calculate the skill difference coefficient between the agents' talents and the tasks' skills for agents I' and tasks J' (Equation 6)
Get the selection coefficient for the agent/task combination based on the dependency priority (Equation 7)
Assign each agent i the task j with the smallest selection coefficient
Calculate the agent's task execution coefficient (Equation 8)
Tasks available to work. J'
Agents with available time and with no assigned tasks I'
Pending Tasks
Dependent Tasks
Rework Tasks
Completed Tasks
Project Tasks J
Project Agents I

Figure 3-3 Process Flow – Task Allocation

For each time step, the system checks if it is a time for a team meeting or a client meeting, and if so, the flow follows the described steps in Section 3.6.4. Figure 3-4 represents the actions happening if it is a team meeting, a client meeting, or a simple task execution. If a team meeting occurs, the task execution coefficient is updated based on the effect of the meeting. If it is a client meeting, there exists the possibility of rework that applies the equations 11,12, and 13. If no meetings occur, the task execution takes place and the remaining time of the task is updated as described in Section 3.6.3.

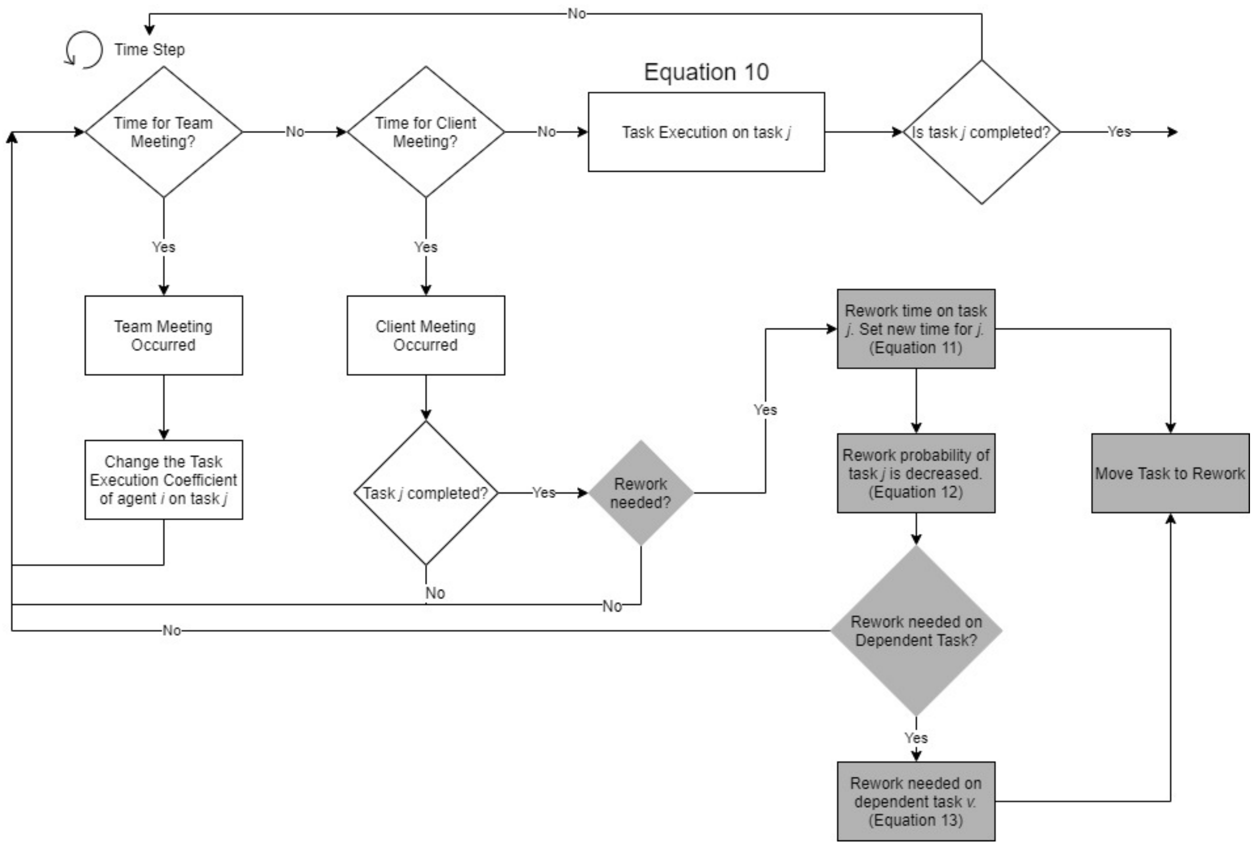

Time Step
Equation 10
Time for Team Meeting?
No
Time for Client Meeting?
No
Task Execution on task j
Is task j completed?
Yes
No
Yes
Team Meeting Occurred
Change the Task Execution Coefficient of agent i on task j
Yes
Client Meeting Occurred
Task j completed?
Yes
Rework needed?
No
No
Yes
Rework time on task j. Set new time for j. (Equation 11)
Rework probability of task j is decreased. (Equation 12)
Move Task to Rework
Rework needed on Dependent Task?
No
Yes
Rework needed on dependent task v. (Equation 13)

Figure 3-4 Process Flow – Task Execution and Meetings

Furthermore, when the in-progress task is completed, the agent's talented are updated, and the agent is available again. If the completed task provides feedback rework to another task, it follows the steps in Section 3.6.4 and applies Equation 14. Additionally, the system checks the tasks which are dependent on the completed task. If all the prerequisite tasks have been completed, the dependent task is now considered as pending – available to performed.

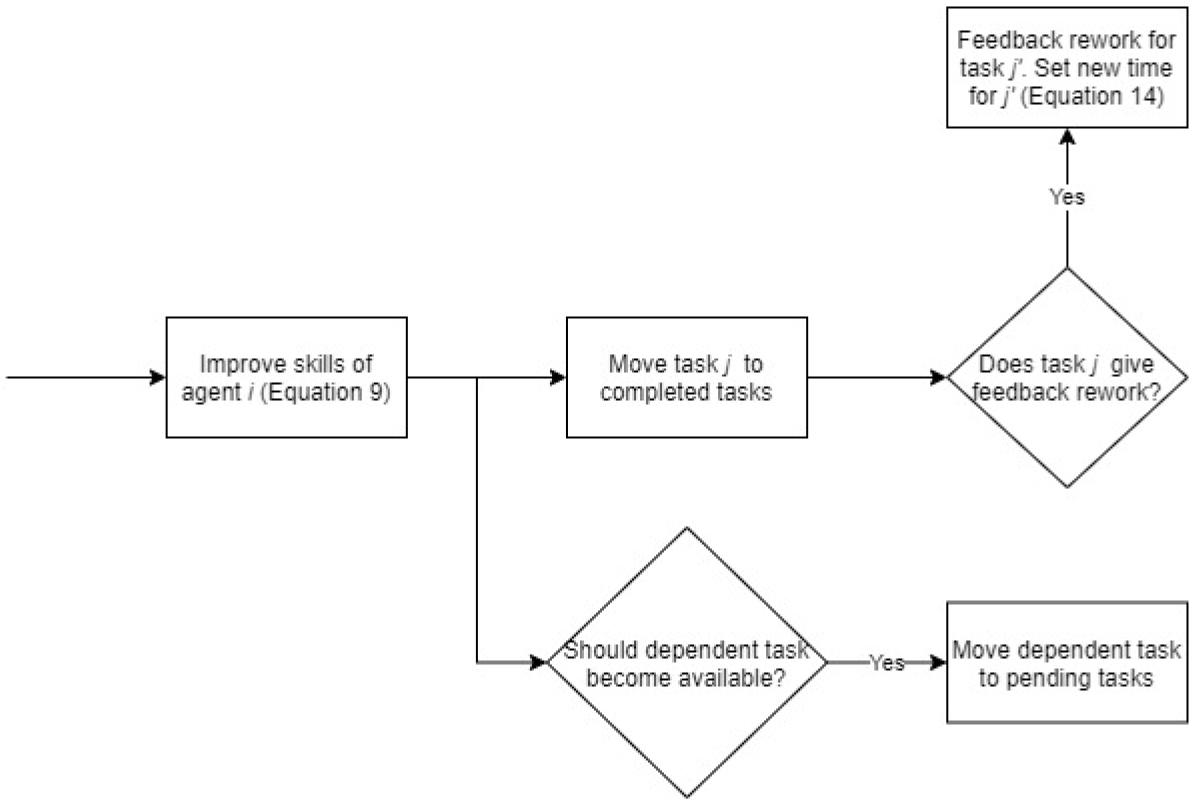

Feedback rework for task *j'*. Set new time for *j'* (Equation 14)
Yes
Improve skills of agent *i* (Equation 9)
Move task *j* to completed tasks
Does task *j* give feedback rework?
Should dependent task become available?
Yes
Move dependent task to pending tasks

Figure 3-5 Process Flow – Task Completion

CHAPTER 4

TESTING, ANALYSIS, AND DISCUSSION

In this chapter, we investigate how the different features affect the model and its results. We also study some popular project management processes and show how they can be simulated using our model. Defining the data of the model is necessary to be able to study the features and understand how each feature affects the results. To compare the results, we need to have suitable tasks and team members. We consider a team of three software developers that have prepared a set of 15 tasks for a new feature in their sales application. The application should allow the users to create orders and select products and their quantities to order. While also viewing a dashboard with reports containing details about their respective orders. Table 4-1 contains the tasks' descriptions and respective times. Each task has a specific identification number and a range of time values (in hours). Next, we define the level of skills required for each task. As explained previously, each task has 3 characteristics each with their weights. Additionally, each task has a priority weight used for the dependency priority only for the prerequisite tasks. Thus, we build Table 4-2.

Description	Task Number	Min Time	Med Time	Max Time
Orders List Page Design	1	20	24	30
Orders List Page Design Implementation	2	16	20	24
Fetch Existing Orders	3	12	15	20
Order Orders by Date	4	2	4	7
Order Creation Page Design	5	6	10	15
Order Creation Page Design Implementation	6	12	15	20
Order Creation Implementation	7	3	5	8
Order Products Selection	8	15	20	22
Order Products Quantity Selection	9	16	20	25
Order Deletion	10	9	12	15
Order Submission	11	4	8	13
Order Invoice Generation	12	13	16	20
Report Generation	13	2	5	8
Order Dashboard	14	7	10	15
Testing	15	20	25	30

Table 4-1 Tasks Description & Times

Task Number	Technical Skill	Managerial Skill	Overall Exp	Technical Weight	Managerial Weight	Exp Weight	Priority Weight
1	8	3	7	0.5	0.2	0.3	0.2
2	9	5	6	0.6	0.2	0.2	0
3	7	5	4	0.4	0.3	0.3	0.2
4	3	4	8	0.1	0.4	0.5	0
5	7	8	8	0.3	0.4	0.3	0.3
6	7	5	8	0.5	0.2	0.3	0.2
7	9	7	8	0.4	0.2	0.4	0.5
8	2	6	6	0.1	0.5	0.4	0.4
9	5	7	4	0.2	0.4	0.4	0.1
10	9	6	6	0.6	0.2	0.2	0
11	3	8	5	0.4	0.3	0.3	0.2
12	5	7	8	0.1	0.5	0.4	0
13	7	8	8	0.2	0.3	0.5	0
14	4	10	4	0.2	0.5	0.3	0
15	6	7	8	0.3	0.3	0.4	0

Table 4-2 Task Requirements

Having defined the requirements, we must consider the rework feature. To do so, we need to estimate the probability of having rework for each task, and the percentage of rework required. Since these values are related to the client frequency meeting f_{mc}, it depends on the manager to estimate these values. The higher the value of f_{mc}, the more likely to get rework since the client's input and feedback comes at a late stage. Usually, this gets easier to estimate as a manager becomes more experienced and has better knowledge of the client. This leads to obtain Table 4-3.

Task Number	**Rework Probability (%)**	**Rework Percentage (%)**
1	25	25
2	15	30
3	15	20
4	30	35
5	50	35
6	30	20
7	25	30
8	5	20
9	15	30
10	15	20
11	30	35
12	25	30
13	15	30
14	35	50
15	20	10

Table 4-3 Task Rework

The final step is setting the DSM for the task dependencies. For our example, there exists multiple dependencies for several tasks that connect and interconnect to represent the work required for each task – based on its description. Table 4-4 represents the dependency DSM. Finally, we specify the team members involved in the project. Table 4-5 Represents the team members and their characteristics. To study the features of our model, it is necessary to study the results of the model with the different features available. By setting/unsetting the features and their values, we are able to see how each feature affects the model.

Task #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-		25/15												
2	70/20	-													
3			-												
4			50/5	-											
5					-		30/15								
6					30/15	-									
7							-	30/25							
8								-					25/15		
9							25/60	10/100	-						
10							50/5	25/15	20/5	-					
11							30/10	40/25	50/5		-		20/25		
12											50/70	-			
13													-		
14														-	
15	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	-

Table 4-4 Task Dependency DSM

First Name	Last Name	Learning-Ability	Tech Skill	Mg Skill	Overall experience	Time Available
John	Smith	7	7	7	5	300
Jane	Johnson	2	4	9	10	300
Kevin	Hart	5	8	4	7	250

Table 4-5 Team Characteristics

4.1 SDC Task Allocation

Our first and most important feature is the Task Allocation Scheme. In order to test it properly within the model, we created a Random Allocation Scheme that simply assigns agents to tasks in a random manner. However, the cooperation feature would influence the allocation. To get comparable results, the settings of the model need to have no effect on task allocation. By turning off the cooperation feature, we are able to better compare between the SDC Allocation and the Random Allocation. Furthermore, the settings consider a working day of 10 hours. The meeting frequencies α_{mc} and α_{me} represent a weekly client meeting and a daily team meeting, respectively. As mentioned in Section 3.6.5, the agent's idle time on task selection $\chi = 1$. Therefore, the settings for our first test are as shown in Table 4-6.

Settings	Status
Client Meetings	ON
$\boldsymbol{\alpha_{mc}}$	44 hours
Cooperation	OFF
Learning	ON
Task Dependency	ON
Dependency Priority	3
Team Meetings	ON
$\boldsymbol{\alpha_{me}}$	10 hours

Table 4-6 Allocation Settings

Using the settings in Table 4-6, we simulated the model on the data specified in Tables 4-1 to 4-5 for a total of 3000 runs on the Random Allocation and another 3000 runs on our SDC Allocation. By normalizing the results with emphasis on the task completion time for each task, we obtained a 7.1% decrease in total task completion time. The total manhours (completed on all 15 tasks) was 7.1% less by using the SDC Allocation instead of Random Allocation.

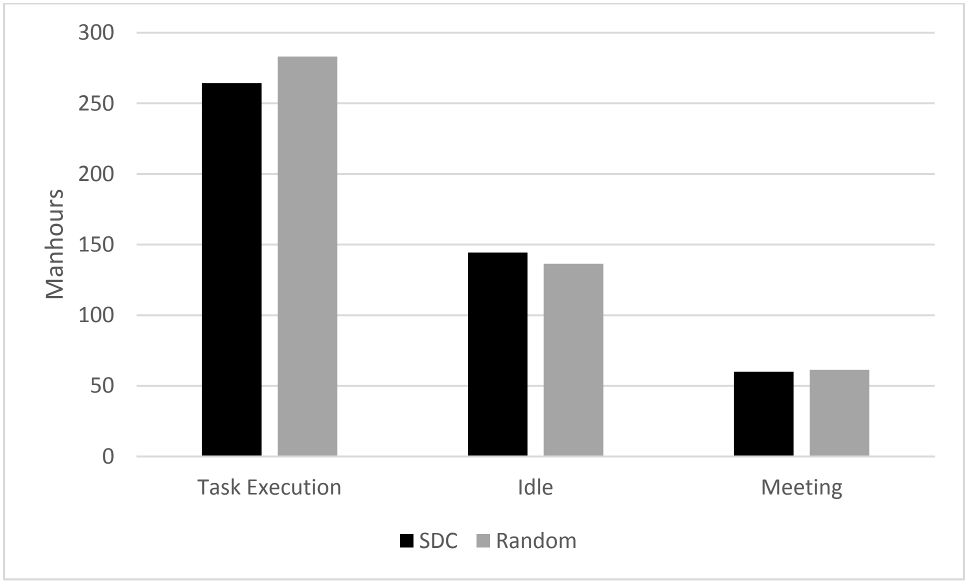


Figure 0-1 SDC Vs Random Agent Effort

Figure 4-1 indicates the values for the Random Allocation and the SDC Allocation in manhours. The total task execution time is significantly lower in the SDC compared to the Random Allocation, indicating that the allocation scheme is better. The idle time is higher in SDC since the agents would finish their tasks quicker, and with no cooperation allowed, they would have to wait for another unallocated task to become available. Additionally, the total meeting time does not change since the meeting will occur every α_{me} hours regardless of the case. It is slightly lower since the project ends faster with SDC allocation than the random allocation. Our biggest advantage is the reduction of the total task time which indicates that less effort is required by the agents to complete a task, given that the agents' talents match the tasks' required skills in a better manner than Random Allocation. Furthermore, the total project time is marginally lower with SDC showing a 2% decrease compared to the Random Allocation. Figure 4-2 represents the total project time for each of SDC Allocation and the Random Allocation. We see a slight decrease for the SDC, this is due to the reduced task execution time for SDC.

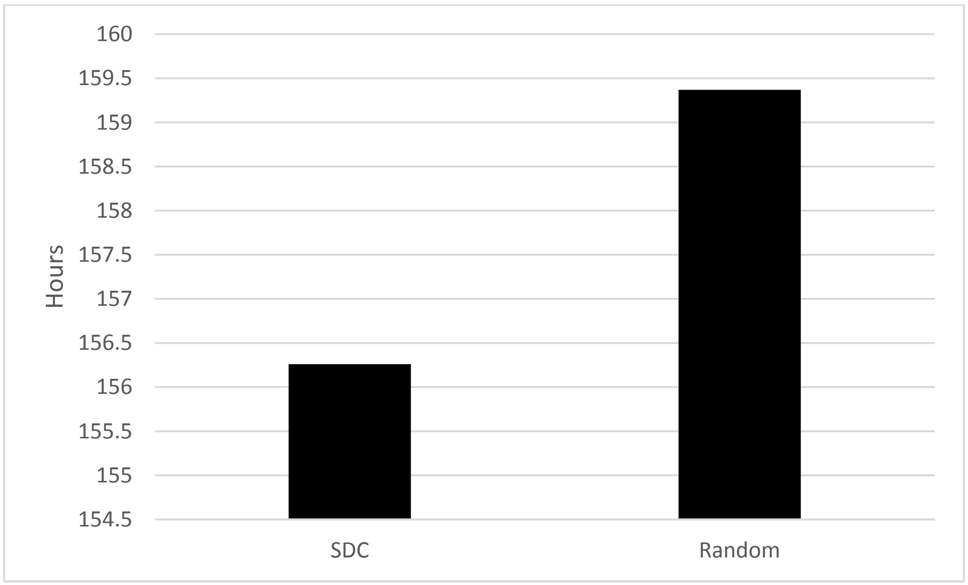


Figure 0-2 SDC Vs Random Allocation Total Project Time

4.2 Agent Learning

In this section, we study how the agent learning feature affects the model. As agents complete more tasks, their talents improve based on Equation 9. To assess this feature with no interference from other features, the settings were defined to only have learning enabled. Table 4-7 shows the settings of this simulation.

Settings	Status
Client Meetings	OFF
$\boldsymbol{\alpha_{mc}}$	-
Cooperation	OFF
Learning	ON
Task Dependency	OFF
Dependency Priority	-
Team Meetings	OFF
$\boldsymbol{\alpha_{me}}$	-

Table 4-7. Agent Learning Settings

With the settings set in Table 4-7, we would be better able to study the agent learning feature. Our focus would only be on the learning and the improvement of the agents, allowing us to assess the benefits and the effects of this feature. Let us look into their improvement rate, and how that influences the task performance time.

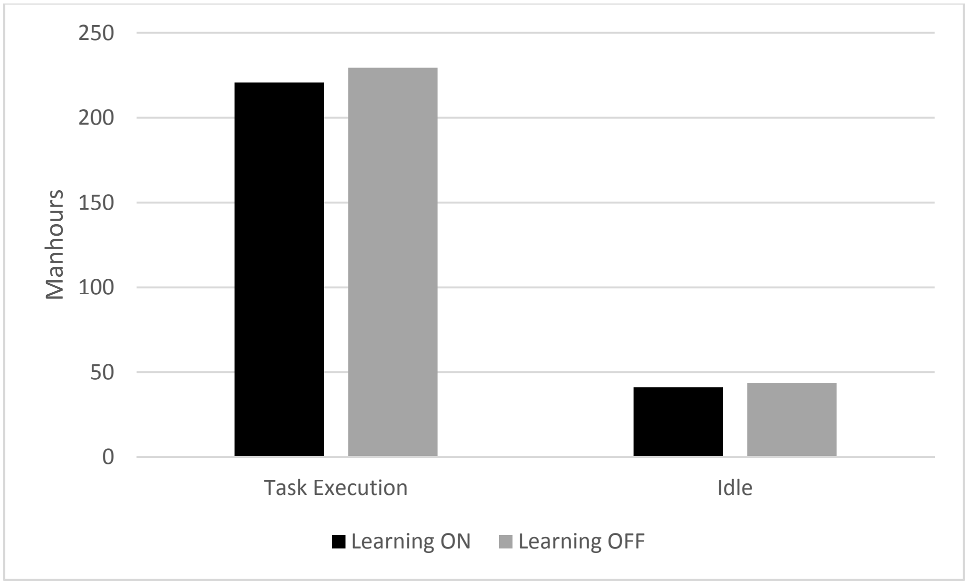


Figure 0-3 Agent Learning Agent Effort

In Figure 4-3, we see how task performance changes with the learning enabled. As the agents are improving with each completed task, they should be reducing their task execution time accordingly. Based on Figure 4-3, we see a 4% decrease in task execution time when the agent learning is enabled compared to the absence of learning. Additionally, we can investigate the agents' improvement and talents after the project. Figure 4-4 shows how each of the agent's talents have improved. Since the agents do not improve with the learning disabled, we can consider the "No Learning" results as their initial talents. Based on this, we built Table 4-8 which contains the improvement rate for each agent on each skill.

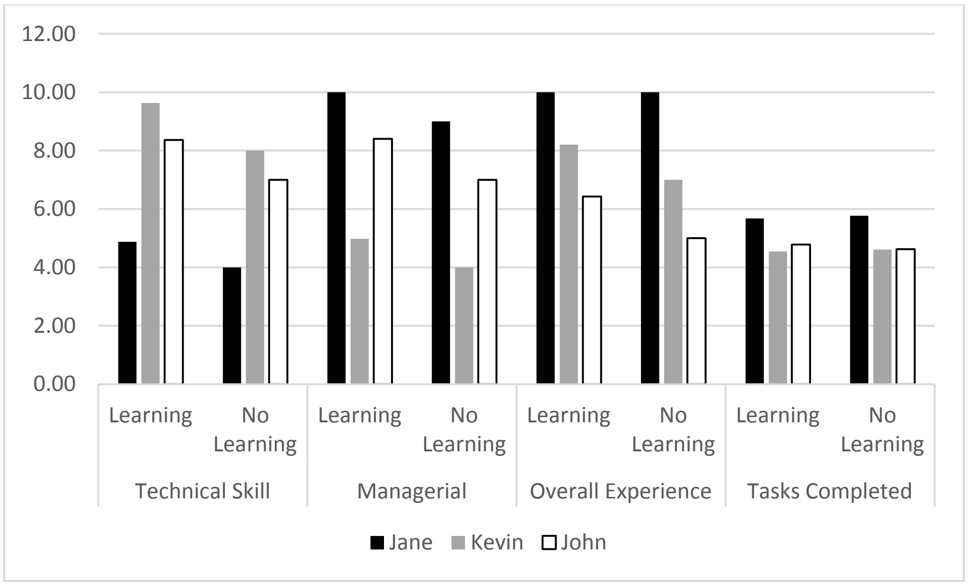


Figure 0-4 Agent Learning – Agent Talents

	Technical Skill			Managerial Skill			Overall Experience		
	No Learning	Learning	(%)	No Learning	Learning	(%)	No Learning	Learning	(%)
Jane	4.00	4.87	21.84	9.00	10.00	11.11	10.00	10.00	0.00
Kevin	8.00	9.63	20.37	4.00	4.98	24.48	7.00	8.21	17.26
John	7.00	8.37	19.51	7.00	8.40	20.06	5.00	6.43	28.56

Table 4-8 Agent Talents

In Table 4-8, we assess the improvement of each agent. Agent Jane for example, has a high managerial talent initially and a low technical talent, and a full score (10/10) in experience. We see no improvement in the experience since it already has a full mark. In the managerial talent, an improvement of 11.11% would lead her to have a full score as well since her initial managerial score was 9/10. Even though her technical improvement percentage is high at around 22%, but the value of increase is lower than the others. This is due to the allocation that would assign Jane on tasks with a high managerial weight, thus leading her to a focused improvement. Regarding agent Kevin, its technical improvement is also high given that the task allocation would focus on assigning Kevin to the more technical tasks considering Kevin's talents. Also, its initial managerial talent is relatively low so its improvement in the managerial aspect would get a high percentage even on a score wise.

4.3 Dependency Priority

In this section, we investigate the dependency priority feature. This feature (discussed in Section 3.5), gives a higher priority to the prerequisite tasks based on their priority weight w_j^p**.** Simply put, the prerequisite tasks would have a value for the priority weight based on how important their completion is to allow work on dependent tasks. By calculating the selection coefficient $\mu_{ijn_iq_i}$ we are using the priority weight and the dependency priority P to allocate the tasks. In this section, we vary P between $P = 1$ and $P = 9$ to see how the allocation would change. The settings for this test are as shown in Table 4-9. We set all the other features to off and leave only the dependency feature on. The results of the simulations are shown in Table 4-9.

Settings	**Status**
Client Meetings	OFF
$\boldsymbol{\alpha_{mc}}$	-
Cooperation	OFF
Learning	OFF
Task Dependency	ON
$\boldsymbol{P}$	-
Team Meetings	OFF
$\boldsymbol{\alpha_{me}}$	-

Table 4-9 Dependency Priority Settings

	$\boldsymbol{P = 1}$	$\boldsymbol{P = 9}$
Task Execution Time (manhour)	234.05	236.90
Idle (manhours)	116.3	100.7
Total Project Time (hours)	115.12	109.82

Table 4-10 Dependency Priority Results

As can be seen in Table 4-10, the total project time is almost identical. The only difference between the two values of P lies between the task execution times and the idle times, respectively. The first difference is in the execution times. Even though the difference is negligible, this increase for $P = 9$ would indicate that even though the task might not be the most suitable or matching for the agent, the agent would be allocated on it due to the dependency priority. In other words, the execution on the tasks would take a bit longer since it would not be the best allocation considering talent-skill matching. However, by having $P = 9,$ the dependent tasks are available to be worked on quicker. Consequently, this is presented in the lower idle time. This lower idle time is due to the higher number of available tasks, in such a way the agents have a higher chance to find an unallocated task to work on.

4.4 Cooperation

In this section, we study the effect of cooperation between agents on the same tasks. By defining the cooperation threshold h, we enabled cooperation on the tasks that have been assigned to an agent $i,$ but with $\omega_{ijn_iq_i} < h$. Before studying this feature, we first need to redefine the tasks' skills and the agents' talents. We set the agents' data as presented in Table 4-11.

First Name	Last Name	Learning-Ability	Technical Skill	Managerial Skill	Overall Exp	Time Available
John	Smith	7	6	6	5	250
Jane	Johnson	2	4	6	7	250
Kevin	Hart	5	9	9	9	250

Table 4-11 Cooperation Agents Talents

One very knowledgeable agent and two relatively weaker agents are used to emphasize the cooperation ability and importance. By having two weaker agents, it is almost certain they would need assistance on their allocated tasks. However, to enforce this cooperation, we also redefine the tasks' skills as presented in Table 4-12. The tasks in Table 4-12 have a higher skill level required than those defined earlier. By doing so, these tasks will require cooperation. Next, we set the settings for the cooperation feature in seen in Table 4-13.

Using the settings in Table 4-13, we study the difference between having cooperation at two different thresholds and having no cooperation. The results are displayed in Figure 4-4.

Task Number	Tech Skill	Mg Skill	Overall Experience
1	8	7	7
2	9	5	6
3	7	5	7
4	3	7	8
5	7	8	8
6	7	7	8
7	9	7	8
8	5	6	6
9	5	7	7
10	9	6	7
11	4	8	5
12	6	7	8
13	7	8	8
14	7	10	7
15	7	7	8

Table 4-12 Cooperation Tasks Skills

Settings	Status
Client Meetings	OFF
$\boldsymbol{\alpha_{mc}}$	-
Cooperation	ON/OFF
$\boldsymbol{h}$	{-1, 0}
Learning	OFF
Task Dependency	OFF
$\boldsymbol{P}$	-
Team Meetings	OFF
$\boldsymbol{\alpha_{me}}$	-

Table 4-13 Cooperation Settings

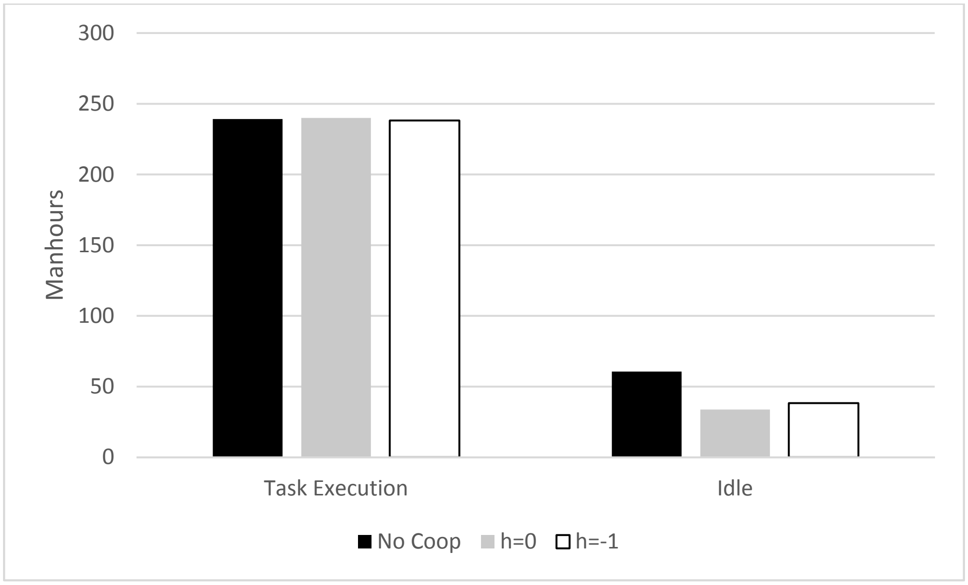


Figure 0-5. Cooperation Agent effort

As seen in Figure 4-5, the task execution time is almost identical between all three cases. The difference can immensely be seen in the idle times. With no cooperation allowed, the agents would have to wait for tasks to become available (due to dependency), and thus their idle time would increase. On the other hand, with cooperation allowed at h=-1, the agents would work together on the tasks while taking into consideration the cooperation threshold h. This is why the idle time is less than the no cooperation case (35% decrease in idle time between no coop to cooperation with $h = -1$). Lastly, if cooperation is allowed with no threshold ($h = 0$), this indicates that any agent i could work on any task j if their selection coefficient is the lowest – even though it might be assigned to another agent. In this case and the scenario considered (with this test's settings), we see a decrease of 45% in the idle time between no cooperation and cooperation with no threshold. This is due to the task allocations that keep all agents working and executing the tasks that are chosen by the selection coefficients.

With such a reduced idle time, we obtain Figure 4-6 which indicates the respective project times for all three cases.

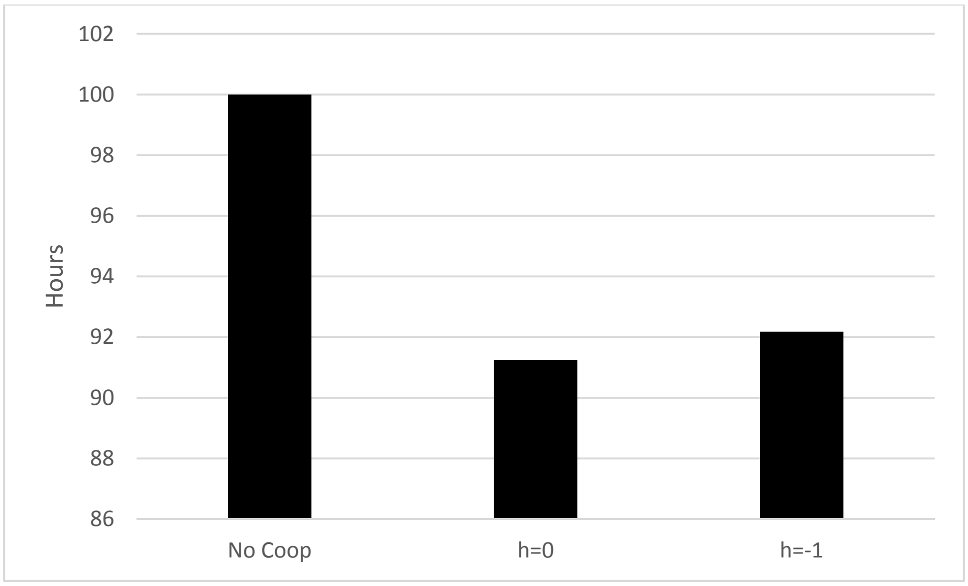


Figure 0-6. Cooperation - Total Project Time

In Figure 4-6, the difference in total project time between no cooperation and cooperation with $h = -1$ is a decrease of around 7%. Furthermore, the 40% decrease in idle time between the no cooperation case and cooperation with no threshold ($h = 0$) is reflected in a 9% decrease in the total project time. These results confirm that for a scenario where most of the tasks are tougher than the agents – have skill levels that are higher than the agents' talents, it is better to have cooperation to reduce the total project time.

As a summary, the task execution time is barely affected since the task execution would be split between two agents. The idle time is lowered since agents would be working at a higher rate since they're not waiting for tasks to become available. And consequently, the project time is reduced since the same amount of work is done in parallel instead of sequentially.

4.5 Client Meeting

In this test case, we are going to focus on the client meeting and rework features. Our team members are those of Table 4-5 (defined originally). Since the client meeting and rework features are related, we assume that the rework probability ρ_{j,k_j} depends on the client meeting frequency α_{mc}. In other words, the less client meetings that occur, the higher the rework probability since the client has not given their feedback for some time and what was completed is more likely to be different than what the client needs. For this reason, we consider three test cases with different client meeting frequencies and different task rework probabilities and rework percentages (defined in Chapter 3).

	$\alpha_{mc} = 22$		$\alpha_{mc} = 44$		$\alpha_{mc} = 75$	
Task Number	Rework Prob	Rework Percentage	Rework Prob	Rework Percentage	Rework Prob	Rework Percentage
1	20	15	30	25	60	50
2	12	20	18	30	40	60
3	15	12	18	20	40	40
4	20	25	30	35	60	70
5	35	25	50	35	80	70
6	22	15	30	20	60	40
7	20	20	25	30	50	60
8	4	12	5	20	30	40
9	12	20	15	30	40	60
10	12	12	15	20	50	40
11	25	25	30	35	65	70
12	22	20	25	30	70	60
13	12	20	15	30	50	60
14	27	30	35	50	70	100
15	15	6	20	10	60	20

Table 4-14 Rework probabilities for client meeting frequencies

Table 4-14 represents the rework probabilities and rework percentages for α_{mc} = 22, α_{mc} = 44, and α_{mc} = 75,respectively . We consider that by having a client meeting often, we would have a lower rework probability and rework percentage. On the other hand, having infrequent client meetings would indicate the client has not given feedback on tasks for a longer time. This is reflected by the increased rework probability and percentage. By focusing on the client meeting alone – with no other feature enabled, we obtained the following results displayed in Figure 4-7.

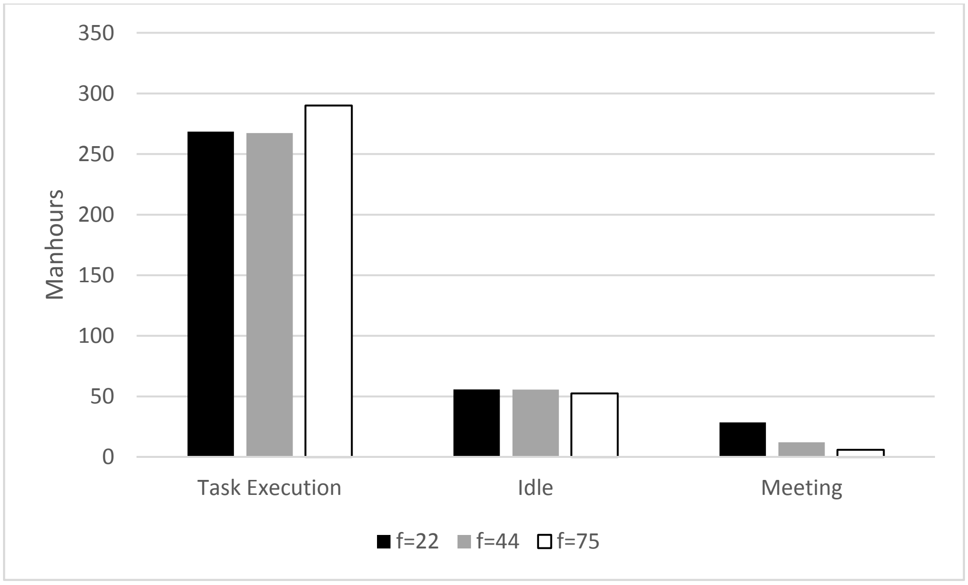


Figure 0-7 Client Meetings Agent Effort

In Figure 4-7, let us first focus on the difference in the task execution time between α_{mc} = 44 and α_{mc} = 75. A frequency of 75 (having a client meeting every 75 hours) is accompanied by the higher percentage of rework. This is reflected in the higher task execution time which is due to the rework after the meeting. Having a medium frequency of meetings with α_{mc} = 44 has an 8.5% decrease in the task execution time compared with α_{mc} = 75. The idle time is almost identical in all three cases since we only have the client meeting feature ON (Cooperation is OFF, Dependency is OFF). Lastly, the meeting times would be the highest with the lowest frequency since a client meeting would take place every 22 hours – within the project the agents and the client would meet several times thus leading to a higher meeting time than the project with a lower client meeting frequency. This is reflected in the total project time seen in Figure 4-8.

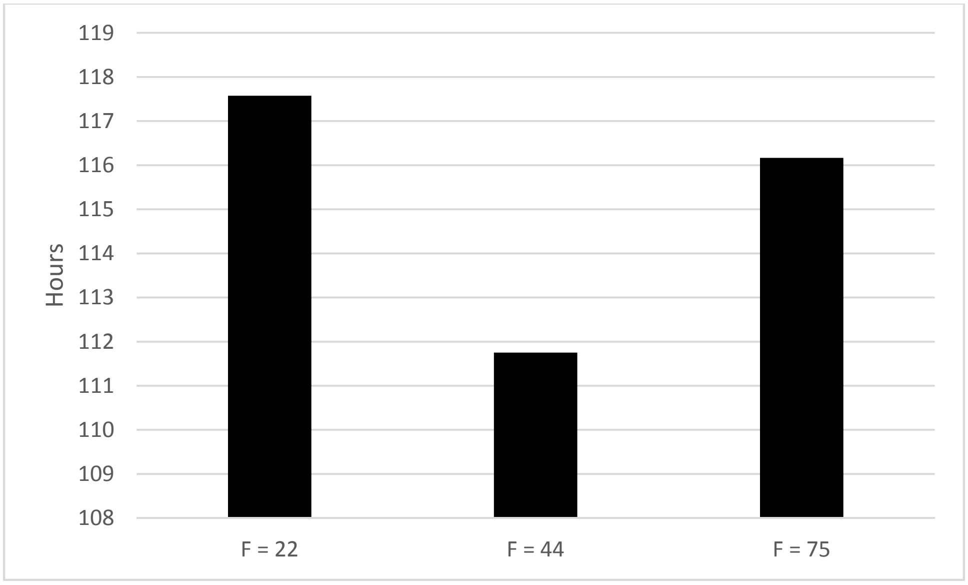


Figure 0-8 Client Meeting total project time

Figure 4-8 represents the total project time in all three cases of α_{mc}. As seen earlier, $\alpha_{mc} = 22$ has the highest meeting in terms of effort, and consequently, this leads to having the highest project time. Second highest, is having too few meetings. With $\alpha_{mc} = 75$, the effort for meeting time is the lowest; however, due to rework and the effort in task execution, it has a higher project time than $\alpha_{mc} = 44$. In a sense, these results show a conversion for the optimized/preferred value of α_{mc}. This relates back to having too few meetings, too many, or just enough.

4.6 Team Meetings

In this section, our focus is on team meetings. We intend to analyze the effect of team meeting frequency on task execution time, as well as the project duration. All the features are turned off except the team meetings. We shall set the team meeting frequency α_{me} for three different cases. We consider $\alpha_{me} = 7$, $\alpha_{me} = 14$, $\alpha_{me} = 30$, indicating that a team meeting happens every 7 hours, 14 hours, and 30 hours respectively.

Effort (Manhour)	**f=7**	**f=14**	**f=30**

Task Execution (manhours)	214.48	218.76	221.37
Idle (manhours)	43.72	44.67	44.70
Meeting (manhours)	40.78	18.47	7.35

Table 4-15 Team Meetings effort

As seen in Table 4-15, for very frequent meetings ($\alpha_{me} = 7$), the execution time is the lowest among the three cases. This is due to having more meetings and with the meetings having a positive effect on the agents' execution coefficients. On the other hand, having less frequent meetings ($\alpha_{me} = 14$) indicates less communication and consequently less improvement on the task execution coefficients (i.e. less advice on task execution through meetings). Even more so, by having a team meeting every 30 hours ($\alpha_{me} = 30$) the task execution effort is the highest among the cases. On the other hand, the effort done by the agents to attend meetings in all three cases is shown to be very high in the more frequent meeting case. With a meeting every 7 hours, agents participate in a meeting very frequently and thus their meeting effort would be high. This effort affects the project time in a direct way. This is shown in Table 4-16 which contains the project times for all three cases. Due to the higher meeting frequency, the project time would be higher. In this sense, having more frequent meetings did reduce the task execution time but on the other hand it caused the project time to rise. The challenge would then be to find the converging frequency of team meetings based on the project and the team.

	f=7	**f=14**	**f=30**
Project Time (Hours)	99.66	93.97	91.14

Table 4-16 Team Meeting Project Time

CHAPTER 5

CASE STUDY

In this chapter we demonstrate our Agent Based Model using two case studies. We can present different PD environments by modifying the simulation settings. By allowing or preventing cooperation, dependency, team meetings and client meetings (or changing their frequencies), we can simulate different PD processes.

5.1 Agile Model

Our first use case is the Agile Model. We consider the same project introduced earlier in Chapter 4, but with some changes to the model settings and a longer task list. This case represents one sprint of the Order Management application. The work in the Agile process consists of iterative sprints which ends by delivering one or more product deliverables. The goal of the Order Management application is to develop a portal for users to be able to use and place orders for different types of products. The user should be able to sign up and login to the application. Then, the user could create an order by going through the available products and selecting what matches its needs. After completing this step, the user should submit the order and obtain a detailed invoice. The application should also contain a page displaying the user's previous orders and their respective pricings. Finally, it should have a dashboard containing the types of orders submitted by the user and a report of the order's products.

Tables 5-1,5-2,5-3,5-4, and 5-5 represent the simulation settings, tasks durations, task execution characteristics, predecessor relationships between tasks using a DSM, and agents, respectively, for the Order Management application. Within a four-week sprint, the simulation settings consider an Agile team that meets every day (thus a meeting frequency of 9 hours); this team would also meet with the client on a weekly basis (every 45 hours). Since dependencies exist, we would set the dependency priority P=2 to give the prerequisite tasks a moderate advantage in the allocation process. Cooperation exists in the Agile model but since the scenario would not recommend unneeded cooperation, we consider the cooperation threshold h= -1. Furthermore, the agents would be learning and developing their skills as they complete tasks. The dependencies consider the testing phase as a task (task 19 in the DSM) and thus it is dependent on all the other tasks and could give rework to any of them.

Setting	Value
Learning	ON
Cooperation	-1
Dependency Priority	2
Meetings	Every 9 hours
Client Meetings	Every 44 hours

Table 5-1 Agile Settings

Description	Task Number	Min Time	Med Time	Max Time
Orders List Page Design	1	20	24	30
Orders List Page Design Implementation	2	16	20	24
Fetch Existing Orders	3	12	15	20
Order Orders by Date	4	2	4	7
Order Creation Page Design	5	6	10	15
Order Creation Page Design Implementation	6	12	15	20
Order Creation Implementation	7	3	5	8
Order Products Selection	8	15	20	22
Order Products Quantity Selection	9	16	20	25
Order Deletion	10	9	12	15
Order Submission	11	4	8	13
Order Invoice Generation	12	13	16	20
Report Generation	13	2	5	8
Order Dashboard	14	7	10	15
Login Page Design	15	8	10	12
Login Page Implementation	16	12	15	18
Sign Up Page Design	17	8	10	12
Sign Up Page Implementation	18	15	20	22
Testing	19	20	25	30

Table 5-2 Order Management App Tasks Times

Task Number	Tech Skill	Mg Skill	XP Skill	Tech Weight	Mg Weigh t	XP Weigh t	Priority Weight	Rewor k	Rework Percentag e
1	8	7	7	0.5	0.2	0.3	0.2	25	25
2	9	5	6	0.6	0.2	0.2	0	15	30
3	7	5	7	0.4	0.3	0.3	0.2	15	20
4	3	7	8	0.1	0.4	0.5	0	30	35
5	7	8	8	0.3	0.4	0.3	0.3	50	35
6	7	7	8	0.5	0.2	0.3	0.2	30	20
7	9	7	8	0.4	0.2	0.4	0.5	25	30
8	5	6	6	0.1	0.5	0.4	0.4	5	20
9	5	7	7	0.2	0.4	0.4	0.1	15	30
10	9	6	7	0.6	0.2	0.2	0	15	20
11	4	8	5	0.4	0.3	0.3	0.2	30	35
12	6	7	8	0.1	0.5	0.4	0	25	30
13	7	8	8	0.2	0.3	0.5	0	15	30
14	7	10	7	0.2	0.5	0.3	0	35	50
15	7	4	8	0.4	0.1	0.5	0.4	25	20
16	6	7	6	0.3	0.4	0.3	0	15	25
17	5	7	7	0.2	0.4	0.4	0.3	25	30
18	8	7	5	0.5	0.3	0.2	0	20	30
19	7	7	8	0.3	0.3	0.4	0.1	20	10

Table 5-3 Order Management App Tasks

Task #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	-		25/15																
2	70/20	-																	
3			-																
4			50/5	-															
5					-		30/15												
6					30/20	-													
7						50/15	-	30/25											
8								-					25/15						
9							25/60	10/100	-										
10							50/5	25/15	20/5	-									
11							30/10	40/25	50/5		-		20/25						
12											50/70	-							
13													-						
14														-					
15															-				
16															50/20	-			
17																	-		
18																	30/30	-	
19	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	-

Table 5-4 Order Management App DSM

First Name	Last Name	Learning-Ability	Tech Skill	Mg Skill	Experience	Time Available
John	Smith	7	7	7	5	200
Jane	Totter	2	4	9	10	200
Luke	Skywalker	5	8	4	7	200

Table 5-5 Order Management App Agents

By running our model for the Agile Model, we were able to obtain results that represent how our model works for an Agile sprint. By considering all the tasks performed within the Product Development phase of the Agile process, and the last task (testing) for the testing phase, we can present an Agile sprint. Table 5-6 presents the total amount of effort needed to complete this sprint, as well as the idle and meeting times. Also, it indicates how the effort was distributed between nominal work and rework on the tasks – nominal being the first time performing the task. These percentages represent the highlight of Agile since they demonstrate a fair distribution between regular and rework task execution times. By having more client involvement in the Agile model, the rework percentage is a fair percentage of 18% . Thus, by including the client more frequently, the impact of rework and rework effort needed would be less than having infrequent meetings.

	Nominal	**Rework**	**Total**
Effort (Manhours)	282.98	62.09	345.07
Percentage (%)	82	18	100
Idle	71	0.00	71
Meeting	77.41	0.00	77.41

Table 5-6 Effort Distribution in Agile

Task Index	Estimated Min Time	Estimated Med Time	Estimated Max Time	Nominal	Rework	Total
1	20	24	30	27.02	6.35	33.37
2	16	20	24	21.34	3.93	25.27
3	12	15	20	15.68	1.36	17.04
4	2	4	7	5.51	1.70	7.21
5	6	10	15	12.22	8.70	20.92
6	12	15	20	17.82	4.17	21.98
7	3	5	8	7.44	4.21	11.65
8	15	20	22	19.98	2.60	22.58
9	16	20	25	19.09	3.14	22.24
10	9	12	15	13.23	1.51	14.74
11	4	8	13	8.11	2.69	10.80
12	13	16	20	16.39	2.28	18.66
13	2	5	8	6.13	1.20	7.33
14	7	10	15	12.16	6.46	18.62
15	8	10	12	11.63	2.46	14.09
16	12	15	18	14.74	2.73	17.47
17	8	10	12	10.36	2.91	13.27
18	15	20	22	19.96	3.71	23.68
19	20	25	30	24.17	0.01	24.18

Table 5-7 Effort for tasks – Agile

Table 5-7 represents the effort (in manhours) on each task. Most of the nominal efforts are around the tasks' medium estimated values. The rework efforts depend on the rework probability and rework percentage of each task – as well as the dependency impact and dependency strength between the dependent tasks. Thus, due to rework we would get a relatively higher effort needed than the original estimated time. Noting that the standard work time is mostly between the medium and max times of the task. The total project time for this Case Study is around 182 hours. This value lies within the expected range of 3 to 4 weeks of work. Additionally, Table 5-8 represents the statistics of the agents before and after the sprint, while also indicating their improvement rates. Knowing that the improvement rates relate to the learning ability of the agent, and its number of tasks completed. The new values of the skills relate directly to the number of tasks completed by each agent as well as their respective learning abilities. Jane for example has improved her managerial skill to a top level of 10/10. Having high levels of managerial and experience talents, her allocation would be on tasks of such needed skills and thus her improvement on the technical aspect would be limited. And due to her having a relatively low learning ability, her improvement in the technical was also a bit low.

		Technical Skill	Managerial Skill	Overall Experience	Time Available	Tasks Completed	Learning-Ability
Jane	**Before**	4.00	9.00	10.00	200.00	0	2.00
	After	5.68	10.00	10.00	62.00	12.33	2.00
	Improv (%)	41.93	11.11	0.00	-	-	-
John	**Before**	7.00	7.00	5.00	200.00	0	7.00
	After	9.67	9.71	7.89	58.34	10.39	7.00
	Improv (%)	38.14	38.67	57.72			
Luke	**Before**	8.00	4.00	7.00	200.00	0	5.00
	After	10.00	6.00	9.58	57.18	10.25	5.00
	Improv (%)	25.00	50.00	36.85			

Table 5-8 Agent Statistics – Agile

5.2 Waterfall Model

In this section, we consider a waterfall model. To show the flexibility of our model, this model contains the same tasks in a sequential manner. A few differences exist between the Agile and Waterfall models. The first difference is in the simulation settings shown in Table 5-10. Since this is a waterfall model, and due to the high dependency within this model, the number of available tasks would be limited at each time step. Thus, to reduce idle times and agents not working, the cooperation threshold h is set to 0. Additionally, the client meeting in the waterfall model happens at the end of the project, but since we would like to demonstrate the rework effect of having no client meeting, we set the client meeting frequency to 99 (the maximum value possible). Due to that, we see the difference in the rework probabilities and percentages due to client meetings – displayed in Table 5-10. We would have a higher rework probability and percentages due to having infrequent meetings and the client's feedback which would affect a higher percentage of each completed task – instead of a rather smaller percentage in the case of Agile.

Setting	Value
Learning	ON
Cooperation Threshold	0
Dependency Priority	2
Meetings	Every 9 hours
Client Meetings	Every 99 hours

Table 5-9 Waterfall Settings

Task Number	Rework	Rework Percentage
1	40	50
2	25	50
3	25	30
4	50	40
5	80	40
6	50	30
7	30	40
8	10	35
9	25	40
10	25	35
11	50	50
12	35	40
13	25	40
14	50	60
15	35	40
16	25	30
17	40	40
18	35	40
19	35	15

Table 5-10 Waterfall Rework Probabilities

Task #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	-																		
2	50/20	-																	
3		30/30	-																
4			30/10	-															
5				50/20	-														
6					30/20	-													
7						50/20	-												
8							50/50	-											
9								30/20	-										
10									25/25	-									
11										50/25	-								
12											30/15	-							
13												15/30	-						
14													30/100	-					
15														20/20	-				
16															50/30	-			
17																20/20	-		
18																	50/20	-	
19	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	-

Table 5-11 Waterfall DSM

Table 5-11 is the DSM representing the relations between the tasks. As seen in Table 5-11, the tasks are sequential in their dependencies (Task 2 depends on Task 1, Task 3 depends on Task 2…). This is done to serve as an extreme case of the waterfall model. The frequency of the client meetings is set to $\alpha_{mc} = 99$ since the client meetings are not frequent in the waterfall model and normally happen at the end of the process. With these settings and tasks characteristics, let us investigate the simulation results.

Waterfall	Nominal	Rework	Total
Task Execution (Manhours)	287.95	70.87	358.82
Percentage (%)	80.25	19.75	100.00
Idle	434.25	0.00	434.25
Meeting	108.28	0.00	108.28

Table 5-12 Effort Distribution in Waterfall

Agile	Nominal	Rework	Total
Task Execution (Manhours)	282.98	62.09	345.07
Percentage (%)	82	18	100
Idle	71	0.00	71
Meeting	77.41	0.00	77.41

Table 5-13 Effort Distribution in Agile

Table 5-12 presents the effort distribution in the Waterfall model compared to the effort distribution in Agile displayed in Table 5-13. The total task execution effort is split between the nominal and rework efforts. We notice a relatively high rework percentage of around 20% (70.87/358.82) in the task execution. This is mainly due to the high rework probability and rework percentages related to the waterfall model and the infrequent client meetings. On the other hand, we see that the rework percentage in the Agile model is relatively smaller with a difference of 1.75%. Second, the high idle time is a consequence of the sequential task relations. Even though cooperation is allowed with no threshold and multiple agents performing the same task, cooperation only happens when the task's skill levels are higher than the agent's talent levels. Consequently, this leads to agents waiting around for tasks to become available as they cannot cooperate on the same tasks with other agents. The difference in idle time represents the biggest difference between the waterfall and agile models. Since the waterfall model goes through each task at a time, it shows an idle time 6 times longer than the agile model. And this idle time affects the total project time to last around 313 hours. A relatively long amount of time, but it reflects the extreme waterfall case.

CHAPTER 6

VALIDATION

In this chapter, we test real projects in our simulation model. The goal is to validate the output of our model for each project based on these two points:

- Task Allocation: the project manager would agree or disagree with the task allocation to the team members.
- Project success : the team members and their times of availability are enough to complete the tasks.

The first step is to meet with project managers (PM) and explain the model's uses and expected outputs. Also , it is necessary to define for them the input file templates and their possible values. By doing so, each PM would have provided the details and suitable data for the tasks (their skill levels, expected times, priority and dependencies), as well as specifying the agents' data (their talents, learning ability, and available time). Then, the PMs need to determine the settings of the simulation for each of their projects. The PM would select the values most suitable for their project for each of the following settings: team meeting and client meeting frequencies, the possibility of cooperation and its threshold, agent learning, and task dependency and dependency priority. Finally, we simulate the project using our proposed model and study the results with the PM to validate the feasibility of the project and its agents, while considering the task allocations.

The first project we investigated is called BBL. The BBL PM had provided a list of 11 tasks ranging between short tasks (4 hours of estimated time for completion) and very long tasks (120 hours), with different required skill levels. Also, the PM provided 4 agents (KH,FT,YA,KA) with different talent levels, learning abilities, and available times. The settings determined were for a two-week sprint as shown in Table 6-1.

Setting	Value
Learning	ON
Cooperation Threshold	-1
Dependency Priority	3
Meetings	Every 10 hours
Client Meetings	Every 45 hours

Table 6-1

By using our model, we obtained a success rate of 96.6% (the agents were able to finish the project given their available times). This basically indicates that the time available for each agent suits the project and is enough for the task completion – even considering the meeting times and idle times. Next ,we investigated the allocations and execution of each task. To do so, we determined the agent with the highest execution time for each task and considered this agent to be the lead on the task (i.e. being the allocated agent), and other agents working on the same task would be considered as cooperating or additional help for task execution. Table 6-2 represents the effort (in manhours) done by each agent for each task.

Task Index	**FT**	**KA**	**KH**	**YA**
1	0.43	0.63	0.00	5.28
2	5.96	11.79	0.00	6.66
3	15.90	1.89	0.00	1.88
4	9.63	5.52	0.00	5.74
5	3.92	6.21	0.00	2.72
6	17.05	5.79	88.00	14.78
7	1.81	3.23	0.00	1.88
8	0.17	16.93	0.00	0.09
9	3.04	4.37	0.00	11.15
10	7.15	5.52	0.00	20.12
11	3.84	1.40	0.00	2.33

Table 6-2 Agent Effort on Task BBL

To explain Table 6-2, each cell represents the effort (manhour) performed by each agent on each task. The agent with the highest execution time is the agent assigned to the task. For example, agent YA is assigned to Task 1 since agent YA has the highest execution time (effort) on Task 1. By comparing with the PM with what effectively happened during the sprint, we got the following comments. Several task allocations were accurate as the agents were assigned to the modeled tasks, but some allocations were not successful. Some tasks required a special skill within the technical skill related to specific knowledge of a software tool that only one agent had. This was the case for tasks 6,8,9 which were allocated to different agents in our model, but had the same agent perform them in the actual scenario. Nonetheless, the PM agreed that task allocations were accurate if not for the mentioned missing skill. As for the order of execution, it was relatively accurate due to the dependencies present.

The second project studied is denoted GEO. GEO is a relatively short sprint of 1 week, consisting of 17 tasks and 2 agents. The tasks and agents differ in their respective skills and talents.

	AH	BG
1	4.45	0.24
2	0.09	4.99
3	3.56	2.54
4	0.01	5.98
5	4.00	0.00
6	7.56	0.65
7	2.08	2.40
8	1.24	0.72
9	0.00	5.00
10	0.03	3.96
11	3.53	1.69
12	1.76	5.96
13	0.77	1.17
14	2.12	0.85
15	2.00	0.00
16	0.79	1.19
17	2.00	0.00

Table 6-3 Agent Effort on Task – GEO

The project's success rate is around 94% - given the times of availability of each agent and the estimated times for each task. In Table 6-3, we can determine the lead agent for each task and the total time for the task's completion. The allocation happens based on the highest execution time on each task, with some tasks requiring cooperation throughout the project. After sharing these results with the PM, we obtained a 70% matching rate between the model's task allocations and the actual task executions. Additionally, all the tasks that required cooperation and were modeled to have cooperation were in fact completed by both agents within the sprint. As for the order of execution, with only one dependency within the project, it was not important for the PM to determine. The PM's feedback for the results was that this simulation model is important to have before the launch of the sprint as it provides a better idea of possible allocations; however, it is hard to confirm the actual time spent on each task after the project's completion. This indicates the importance of having this simulation model before the launch of the project or sprint as means to give a better guidance for the task allocation and execution process.

The third project to consider is denoted MDS. This is a long running project split into regular sprints. The PM provided 9 long tasks with different skill levels for the current running sprint of 3 weeks. Also, the PM determined the 5 available agents with their talents and availability times.

	AK	AC	JK	JA	RB
1	0.00	6.51	11.31	0.78	0.46
2	0.00	0.93	10.67	0.85	0.56
3	0.01	1.21	1.29	1.08	1.28
4	0.00	0.00	0.00	0.00	9.00
5	0.00	0.04	0.07	0.01	8.18
6	0.01	0.85	1.56	1.49	0.89
7	0.02	5.02	8.40	41.77	11.05
8	0.01	1.88	2.91	2.39	2.47
9	54.37	33.38	11.70	8.95	16.80

Table 6-4 Agent Effort on Task – MDS

This sprint has a success rate of 75%. This is mainly due to the lack of time availability by the agents, and the high estimated times of completions for the tasks. In Table 6-4, the results determine the agents and their efforts in on each task while indicating which agent would most likely be assigned to each task. After reviewing the results, the PM expressed that the allocations are accurate for most of the tasks, especially with the individual tasks (such as Tasks 4 and 5 assigned to agent RB), and with the collaborative tasks with 1 agent leading the task execution (Task 1 with agent JK leading, and Task 7 with agent JA leading). On the other hand, the task allocation for agents AK and AC was not very accurate. Agents AK and AC are project managers and will most likely be spending their available times across multiple tasks cooperating with other agents instead of concentrating on 1 task specifically (like agent AK's allocation on task 9). Despite that, the PM was satisfied with results of the simulation and indicated that the model would help determine the feasibility of the sprint and the risk of being late on the project – since this is one of the biggest problems faced in this project.

Our next project is denoted MKT. This project represents a specialized marketing unit that works on multiple projects implementing their own specialization. The PM had provided a set of 15 tasks with different skills needed and a set of 3 agents with various talent levels. The tasks are lengthy with an average of 3 days of estimated time of execution, but with no dependency among them since they could belong to different projects.

	CA	**GG**	**MA**
1	0.00	0.00	25.67
2	2.48	25.49	60.63
3	0.00	0.07	27.06
4	0.03	5.26	11.02
5	57.12	84.00	56.42
6	0.01	3.04	12.54
7	1.38	30.03	5.07
8	0.41	5.67	34.51
9	0.00	32.51	0.00
10	6.45	53.99	27.35
11	0.02	35.15	15.81
12	60.01	0.00	0.00
13	0.01	2.72	1.94
14	0.21	4.76	0.42
15	50.36	2.21	1.96

Table 6-5 Agent Effort on Tasks – MKT

The task distribution seems straight forward. Some tasks are set for one agent with no cooperation and without the possibility of another agent being allocated on the task. Task 1 for example is set for agent MA, task 9 is assigned to agent GG and task 12 is allocated to agent CA alone. Some other tasks are allocated to a lead agent but with the possibility of cooperation. This is reflected mostly in tasks 2,10,11 that are shared between agents GG and MA. Furthermore, task 5 required effort from all the agents due to its extended time. Based on the PM’s feedback, the simulation model provided an accuracy of task allocation of 80%. Also, the cooperation indicated in the results was accurate since several tasks were completed by two agents at the same time.

Finally, we investigate our fifth project denoted VLO. VLO is a project based on monthly sprints, during which a set of tasks are assigned and completed by 3 agents. For our simulation, the PM provided 17 tasks with various requirements and estimated times, as well as their respective dependencies.

	CH	CA	TN
1	6.71	1.37	3.27
2	0.00	0.00	7.00
3	0.00	0.17	24.17
4	0.59	2.25	2.17
5	0.26	1.16	1.59
6	0.62	6.74	3.58
7	1.25	1.30	2.07
8	5.96	6.69	4.05
9	0.23	3.52	11.44
10	0.07	6.46	1.54
11	3.55	3.01	1.36
12	1.00	0.00	0.00
13	3.00	0.00	0.00
14	11.34	1.58	2.22
15	0.58	11.55	3.66
16	0.00	23.67	0.00
17	29.87	0.00	0.00

Table 6-6 Agent Effort on Tasks – VLO

The results in Table 6-6 indicate that some tasks are very specific to one agent, whereas other tasks are shared by multiple agents. Tasks 2,3,12,13,16,17 are each assigned to one agent with no need for cooperation or the possibility of another allocated agent. On the other hand, some tasks require cooperation or could be assigned to different agents such as tasks 6-11. By presenting these results to the PM, they indicated that most of the allocations were accurate. However, agent CH has a different role on the project which prevented it from working on all the tasks that the simulation allocated to it. Nevertheless, the PM was satisfied with the results, especially that their sprint obtained a 97% success rate.

To summarize, the five projects studied indicated a good allocation strategy – similar to what the team members actually worked on. All five PMs indicated that the tool or simulation model was very useful when launched before the start of the sprint as it indicated guidelines and was accurate enough to help in their project's management. Some PMs indicated that the lack of roles affects the simulation and the results as some more roles need to be considered. Overall, the task allocation and task execution, along with the cooperation and meeting effects, as well as learning and dependency, proved to be very interesting for the PMs. The company to which these projects is considering developing a tool to help the PMs do their roles in a more advanced and optimized way.

CHAPTER 7

CONCLUSION

Improving project management in the product development process will always be a challenging issue. Along with the different methodologies, the project manager would always face a challenge to make the best decisions for the project to succeed within time and budget. Thus, arises the need for a simulation tool that would provide a better guidance and advice for the project manager, in a way to assist them to avoid project failure. The simulation model in this thesis serves as a tool with different settings and features that adapt to the project's actual circumstances and contributes with its features to help the project manager obtain a better view of the project's status.

Starting with an allocation strategy, along with a task execution estimation, our model focused initially on enhanced task allocation to improve project execution and reduce development time. Then, the model introduced the learning feature to study how an agent would improve by completing tasks, and how their skills would evolve over the course of the project. Next, by integrating cooperation, the model handled the agents that needed support to finish their allocated tasks. Furthermore, communication and negotiations were also covered by implementing team meetings and client meetings in which these two forms of interactions take place. Additionally, the model investigates task dependency and how a task could have multiple prerequisite tasks. Finally, the concept of rework was also modeled either by rework requested by the client, or rework due to feedback from other tasks. By applying sensitivity testing for each of these features, we were able to demonstrate the effectiveness and impact of each feature. All these features put together allowed us to model a product development flow and consider its most important settings to get an output that represented a map for the task distribution within the project.

Furthermore, the two case studies considered allowed us to represent the flexibility of the model which spans the spectrum from the traditional waterfall project management model to the most iterative Agile methodology. Finally, our validation was done on real-life software development projects in which project managers provided their input data and studied the output of the model, agreeing to the allocation strategy and the results of the model.

This model presents a way to improve resource allocation, decide the frequency of team meetings, simulate employee learning, consider the effect of rework, choose when cooperation on the tasks is needed (and beneficial), and map task dependencies. By combining all these, the project manager which uses this model would get a better view of what to expected during their PD process.

However, this model developed has some limitations and thus opportunities for improvement. The first limitation is the number of tasks considered in Netlogo. Since it is a two-dimensional plane, having too many tasks would lead to multiple tasks in the same coordinates which would prevent the agent from finding its allocated task. We could consider using a less visual and a more technical coding language such as Java since our primary concern is the output report. Another limitation is the role played by the agents. This point was raised by one of the PMs during validation, where two agents were PMs and should not be allocated to just one task until its completion. This point is to be taken into consideration, possibly developing additional roles that would not take part in the task allocation, rather oversee the completion of tasks. An additional feature to consider is the dynamic task creation. This would be used to present an urgent task that was created while the project was in progress – such as client asking for a new feature during a client meeting. The solution here would be to restart the allocation. With the new tasks added during the client meeting, it would be enough to restart the initial task allocation on the current available tasks. With the tasks having a higher priority, they are likely to be allocated directly. Lastly, the model focuses on availability to allocate the tasks, but another feature to consider is queueing or saving tasks for a specific agent even though the agent is not available. This would simulate reserving tasks for the most suitable agent until the agent is available. To do so, the model would have to preserve the initial allocations based on the best score for each task/agent combination. That way, upon completing the task, the agent would check what has already been reserved for it to be its next task based on the selection coefficient.

REFERENCES

Abdelsalam, H. M., & Bao, H. P. (2007). Re-sequencing of design processes with activity stochastic time and cost: An optimization-simulation approach. *Journal of Mechanical Design*, *129*(2), 150-157.

Acuña, S. T., & Juristo, N. (2004). Assigning people to roles in software projects. *Software: Practice and Experience*, *34*(7), 675-696.

Acuña , S. T., Juristo, N., & Moreno, A. M. (2006). Emphasizing human capabilities in software development. *IEEE software*, *23*(2), 94-101.

Allen, T. J. (2007). Architecture and communication among product development engineers. *California Management Review*, *49*(2), 23-41.

Al Hattab, M., & Hamzeh, F. (2016). Analyzing design workflow: An agent-based modeling approach. *Procedia engineering*, *164*, 510-517.

Awwad, R., Asgari, S., & Kandil, A. (2014). Developing a virtual laboratory for construction bidding environment using agent-based modeling. *Journal of Computing in Civil Engineering*, *29*(6), 04014105.

Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. *arXiv preprint arXiv:1205.6904*.

Bergner, Y., Andrews, J. J., Zhu, M., & Gonzales, J. E. (2016). Agent-Based Modeling of Collaborative Problem Solving. *ETS Research Report Series*, *2016*(2), 1-14.

Browning, T. R., & Eppinger, S. D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE transactions on engineering management*, *49*(4), 428-442.

Cooper, R. G., & Sommer, A. F. (2016). The agile–stage-gate hybrid model: a promising new approach and a new research opportunity. *Journal of Product Innovation Management*, *33*(5), 513-526.

Cooper, S., & Taleb-Bendiab, A. (1998). CONCENSUS: multi-party negotiation support for conflict resolution in concurrent engineering design. *Journal of Intelligent Manufacturing*, *9*(2), 155-159.

Chien, T. H., Lin, Y. I., & Tien, K. W. (2013). Agent-based negotiation mechanism for multi-project human resource allocation. *Journal of Industrial and Production Engineering*, *30*(8), 518-527.

Denkena, B., Dittrich, M. A., & Winter, F. (2017). Competence-based personnel scheduling through production data. *Procedia CIRP*, 63, 265-270.

Dinu, A. (2016). *Project risk management - reasons why projects fail*. Bucharest: Romanian Society for Quality Assurance.

Ernst, H., Hoyer, W. D., & Rübsaamen, C. (2010). Sales, marketing, and research-and-development cooperation across new product development stages: implications for success. *Journal of Marketing*, *74*(5), 80-92.

Fishman, G. S. (2013). *Discrete-event simulation: modeling, programming, and analysis*. Springer Science & Business Media.

Garcia, R. (2005). Uses of agent-based modeling in innovation/new product development research. *Journal of Product Innovation Management*, *22*(5), 380-398.

Hoegl, M., & Weinkauf, K. (2005). Managing task interdependencies in Multi-Team projects: A longitudinal study. *Journal of Management Studies*, *42*(6), 1287-1308.

Johnson, J., Wu, H., Sifleet, T., & Fathianathan, M. (2008). A Computational Method for Task Allocation and Coordination in a Distributed Design Environment. In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 595-605). American Society of Mechanical Engineers Digital Collection.

Kandemir, C., & Handley, H. A. (2019). Work process improvement through simulation optimization of task assignment and mental workload. *Computational and Mathematical Organization Theory*, *25*(4), 389-427.

Klügl, F., & Bazzan, A. L. (2012). Agent-based modeling and simulation. *AI Magazine*, 33(3), 29-29.

Kusumasari, T. F., Supriana, I., Surendro, K., & Sastramihardja, H. (2011, July). Collaboration model of software development. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics* (pp. 1-6). IEEE.

Kusiak, A., & Wang, J. (1994). Negotiation in engineering design. *Group Decision and Negotiation*, 3(1), 69-91.

Kvan, T. (2000). Collaborative design: what is it?. *Automation in construction*, 9(4), 409-415.

Le, Q., & Panchal, J. H. (2011). Modeling the effect of product architecture on mass-collaborative processes. *Journal of Computing and Information Science in Engineering*, *11*(1), 011003.

Leithold, N., Haase, H., & Lautenschläger, A. (2016). Cooperation in new product development: An analysis of small technology-based firms. *The International Journal of Entrepreneurship and Innovation*, *17*(1), 5-14.

Levy, F. K. (1965). Adaptation in the production process. *Management Science*, *11*(6), B-136.

Maier, J. F., Wynn, D. C., Biedermann, W., Lindemann, U., & Clarkson, P. J. (2014). Simulating progressive iteration, rework and change propagation to prioritise design tasks. *Research in Engineering Design*, *25*(4), 283-307.

Moe, N. B., Dingsøyr, T., & Rolland, K. (2018). To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large-scale agile software development.

Moenaert, R. K., Caeldries, F., Lievens, A., & Wauters, E. (2000). Communication flows in international product innovation teams. *Journal of Product Innovation Management: AN INTERNATIONAL PUBLICATION OF THE PRODUCT DEVELOPMENT & MANAGEMENT ASSOCIATION*, *17*(5), 360-377.

Ngo-The, A., & Ruhe, G. (2009). Optimized resource allocation for software release planning. *IEEE Transactions on Software Engineering*, *35*(1), 109-123.

Nissen, M. E., & Levitt, R. E. (2004). Agent-based modeling of knowledge dynamics. *Knowledge Management Research & Practice, 2*(3), 169-183.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization science*, 5(1), 14-37.

Olfati-Saber, R., Fax, J. A., & Murray, R. M. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, *95*(1), 215-233.

Panchal, J. H. (2009). Agent-based modeling of mass-collaborative product development processes. *Journal of Computing and Information Science in Engineering, 9*(3), 031007.

Pinto, J. K., & Mantel, S. J. (1990). The causes of project failure. *IEEE transactions on engineering management*, *37*(4), 269-276.

Salhieh, S. E., & Monplaisir, L. (2003). Collaboration planning framework (CPF) to support distributed product development. *Journal of Intelligent Manufacturing, 14*(6), 581-597.

Scott, M. J., & Antonsson, E. K. (1996, August). Formalisms for negotiation in engineering design. In *ASME DETC/CIE Design Theory and Methodology Conference*.

Siebers, P. O., Macal, C. M., Garnett, J., Buxton, D., and Pidd, M. (2010). Discrete-event simulation is dead, long live agent-based simulation! *Journal of Simulation, 4*(3), 204-210.

Simpeh, E. K., Ndihokubwayo, R., Love, P. E., & Thwala, W. D. (2015). A rework probability model: a quantitative assessment of rework occurrence in construction projects. *International Journal of Construction Management*, *15*(2), 109-116.

Sliger, M. (2011). Agile project management with Scrum. *Project Management Institute*.

Sommer, A. F., Hedegaard, C., Dukovska-Popovska, I., & Steger-Jensen, K. (2015). Improved product development performance through Agile/Stage-Gate hybrids: The next-generation Stage-Gate process?. *Research-Technology Management*, *58*(1), 34-45.

Stobrawa, S., Denkena, B., Dittrich, M. A., & Jenkner, I. (2018, November). Simulation-Based Personnel Planning Considering Individual Competences of Employee. In *Congress of the German Academic Association for Production Technology* (pp. 613-623). Springer, Cham.

Sweetser, A. (1999, July). A comparison of system dynamics (SD) and discrete event simulation (DES). In *17th International Conference of the System Dynamics Society* (pp. 20-23).

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard business review*, *64*(1), 137-146.

Ulrich, K. T., & Eppinger, S. D. (2012). Concept selection. *Product Design and Development, 5th ed. Philadelphia: McGraw-Hill/Irwin*, *1*, 145-161.

Wright, T. P. (1936). Factors affecting the cost of airplanes. *Journal of the aeronautical sciences*, *3*(4), 122-128.

Yassine, A. A., Whitney, D. E., & Zambito, T. (2001). Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM).

Zankoul, E., Khoury, H., & Awwad, R. (2015). Evaluation of agent-based and discrete-event simulation for modeling construction earthmoving operations. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction* (Vol. 32, p. 1). IAARC Publications.

APPENDIX A

Parameter	Definition
A	Agents
T	Tasks
l	Agent Learning Ability
q_i	Last task completed by agent i
$\Delta_{ijn_iq_i}$	Skill Difference Coefficient
Z_i	Available time for agent i
w_j^m	Managerial weight
$A_{i,r}^m$	Managerial talent at time r
T_i^m	Managerial skill required
w_j^t	Technical weight
$A_{i,r}^t$	Technical talent at time r
T_i^t	Technical skill required
w_j^l	Overall experience weight
$A_{i,r}^l$	Overall experience at time r
T_i^l	Overall experience required
w_j^p	Priority weight
P	Priority Factor
$\mu_{ijn_iq_i}$	Selection Coefficient
m	Meeting Factor
h	Cooperation Threshold
n	Number of tasks completed by an agent
α_{me}	Team Meeting Frequency
β_t	Team Meeting duration
α_{mc}	Client Meeting Frequency
β_c	Client Meeting Duration
$\omega_{ijn_iq_i}$	Task Execution Coefficient
k	Iteration Number on task
r	Time
$\rho\ _j$	Rework Probability of task – due to client meeting
δ_j	Rework Percentage of task– due to client meeting
θ_{jv}	Dependency relation between task j and task v
τ_{jv}	Dependency Strength between task j and task v
ε_{jv}	Dependency Impact between task j and task v
$t_{jrn_ik_jq_i}$	Remaining Time on task j at time r at task Iteration k - considering the agent's last task and its number of complete tasks
Y	Maximum number of agents on task

χ	Idle Time on task selection

APPENDIX B

Example of a Task Details CSV File:

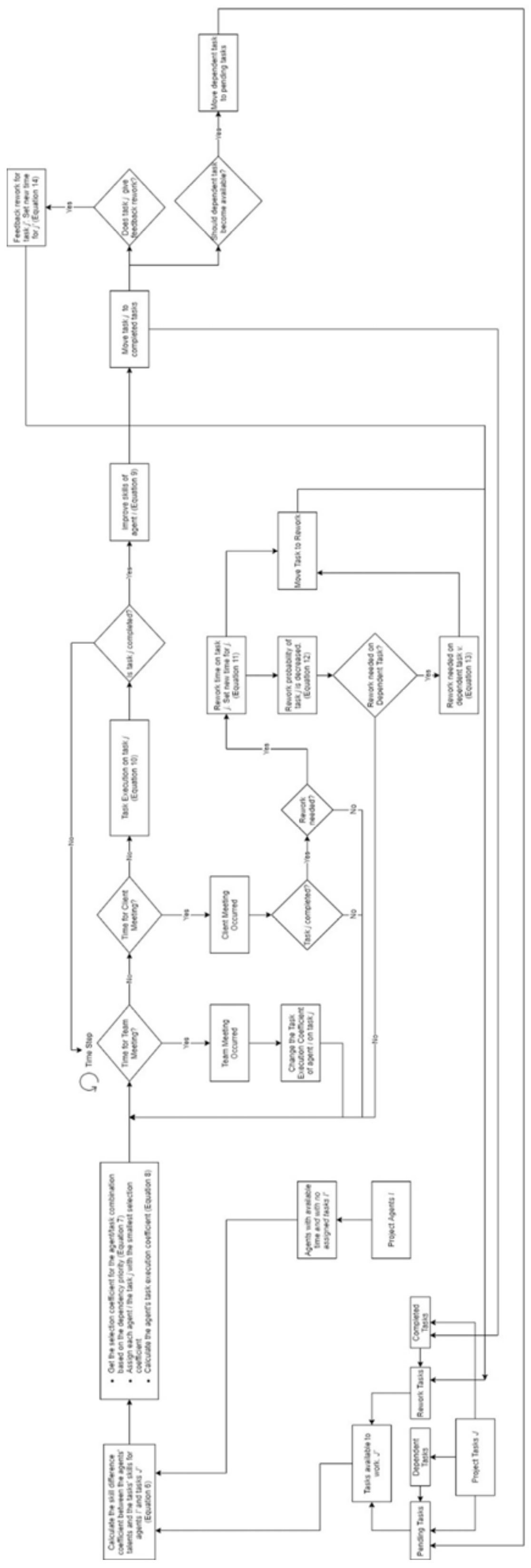
Description	Task Number	Min Time	Med Time	Max Time	Tech Skill	Mg Skill	Experience	Tech Weight	Mg Weight	Exp Weight	P Weight
Task 1	1	4	4	4	6	6	6	0.35	0.35	0.3	0.2
Task 2	2	24	24	24	4	6	4	0.3	0.4	0.3	0

Description	Rework	Rework Percentage	Task Dependency	Dep Strength	Dep Impact	Blocking Dep
Task 1	25	25				
Task 2	15	30	1	20	10	1

Example of an Agent Details CSV File

First Name	Last Name	Learning-Ability	Technical Skill	Managerial Skill	Overall Experience	Time Available
John	Smith	7	7	7	5	250
Jane	Totter	2	4	9	10	250
Luke	Skywalker	5	8	4	7	250

APPENDIX C



Calculate the skill difference coefficient between the agents' talents and the tasks' skills for agents i' and tasks J' (Equation 6)
Get the selection coefficient for the agent/task combination based on the dependency priority (Equation 7)
Assign each agent i the task j with the smallest selection coefficient
Calculate the agent's task execution coefficient (Equation 8)
Time Step
Time for Team Meeting?
Yes
No
Team Meeting Occurred
Change the Task Execution Coefficient of agent i on task j
Time for Client Meeting?
Yes
No
Client Meeting Occurred
Task j completed?
Yes
No
Rework needed?
Yes
No
Task Execution on task j (Equation 10)
Is task j completed?
Yes
No
Rework time on task j. Set new time for j (Equation 11)
Rework probability of task j is decreased (Equation 12)
Rework needed on Dependent Task?
Yes
No
Rework needed on dependent task v. (Equation 13)
Move Task to Rework
Improve skills of agent i (Equation 9)
Move task j to completed tasks
Does task j give feedback rework?
Yes
Feedback rework for task j. Set new time for j (Equation 14)
Should dependent task become available?
Yes
Move dependent task to pending tasks
Agents with available time and with no assigned tasks i'
Project Agents I
Completed Tasks
Rework Tasks
Tasks available to work J'
Dependent Tasks
Pending Tasks
Project Tasks J