MULTI-AGENT BASED DYNAMIC SCHEDULING SYSTEM FOR MANUFACTURING

G.B. Prabash Darshanapriya

168280B

Degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

April 2019

MULTI-AGENT BASED DYNAMIC SCHEDULING SYSTEM FOR MANUFACTURING

G.B. Prabash Darshanapriya

168280B

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

April 2019

Declaration

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of Student

G.B. Prabash Darshanapriya

Signature of Student: Date:

The above candidate has carried out research for the Master's Dissertation under my supervision.

Name of Supervisor

Prof. A.S. Karunananda

Signature of Supervisor: Date:

Acknowledgements

I would like to express my sincere gratitude to my supervisor Prof. A.S. Karunananda for providing his invaluable guidance, comments and feedback throughout the entire project. I would also like to acknowledge him for constantly motivating me to work harder to make this project a success.

Furthermore, I would like to thank my parents, fellow colleagues, and all the lecturers of the Department of Computational Mathematics for the help and support they have given me over the course of this project.

Abstract

Manufacturing scheduling is considered one of the hardest scheduling problems due to its highly dynamic and uncertain nature. The existing approaches for dynamic scheduling with machine learning techniques require a large amount of past-data to be analysed, which results in a substantial amount of time taken to generate schedules.

This study aims to discuss the DynoSchedule system in resolving this highly complex scheduling problem in manufacturing organizations with the help of Multi Agent Technology. In the developed system, depending on the structure of the organization, Agents are generated dynamically for handling of Orders, Machinery (Work-centres). Each of these Agents communicate in an advanced market-like negotiation mechanism considering different factors and try to schedule operations of an order while meeting the required constraints in a greedy manner. However, there's a Manager Agent who oversee the communication and prioritize the requests by evaluating a set of criteria. In addition, the DynoSchedule system introduces the novel concept of Prioritized-Adaptive Scheduling mechanism, an extension to the existing Adaptive Scheduling algorithm, alongside the market-like negotiation mechanism, which makes dynamic scheduling more efficient and effective. The developed DynoSchedule system has been critically evaluated by comparing it with a dataset acquired from a different scheduling system that uses a combination of manual and dynamic scheduling to solve issues that arise due to planned and unplanned interruptions on work centres, or part unavailability. Various indicators such as the percentage of orders to-be-completed on-time, the percentage of tardy orders, work centre availability, Overall Equipment Effectiveness (OEE) and the amount of time taken for the dynamic scheduling process, were considered when evaluating the system. From the obtained results, it was evident that the DynoSchedule system delivers well in terms of the number of orders delivered on-time, the work centre utilization as well as the OEE, providing impressive results.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Prolegomena

Scheduling in manufacturing is the method of assigning resources, arranging and optimizing workloads of the manufacturing process according to a process plan. Scheduling is used to determine the most apt time to perform operations taking into consideration the relationships between manufacturing processes and the capacity restrictions of the shared resources.

Scheduling is widely studied in various research domains such as Genetic Algorithms, Neural Networks, Simulated Annealing and Fuzzy Logic due to its highly combinatorial aspects (NP-hard) as well as the practical efficacy for industrial applications [1]. These types of traditional analytical or heuristic approaches encounter various concerns when applied to dynamic industrial applications since they use simplified models and central computing units to carry out computational tasks which make them essentially centralized and inefficient. This in turn render such methods quite rigid and incapable of addressing the highly dynamic nature of real-world production environments.

As opposed to these traditional methods, Multi Agent (MA) approach has been given a major focus when considering the scheduling problems in the recent past, since state-of-the-art manufacturing systems demand decentralization, distribution, autonomy as well as flexibility to cope with the complex and dynamic real-world scenarios [2]. Per se, the main purpose of this study is to provide an intelligent Multi-Agent based scheduling system for the Constraint-Based Scheduling tasks of an industrial manufacturing organization.

1.2. Aims & Objectives

The aim of this study is to develop a computer-based solution to the highly complex and dynamic constrained-based scheduling problem in Manufacturing organizations.

In addition, following listed are the major objectives related to the research:

- Critical study of Multi-Agent Systems and how it has been used in different industrial applications.
- Critical study of the literature on how to use a Multi-Agent based approach to solve the dynamic production scheduling problem.

- Development of a fully functional solution using Multi-Agent technology as the core of the system.
- Critical evaluation of the implemented system using appropriate Key Performance Indicators (KPIs) and comparative analysis of Multi-Agent Approach with traditional approaches.
- Writing a research paper and a conference paper on the project.
- Producing the final documentation.

1.3. Background & Motivation

A constraint-based scheduling problem decides on when to carry out different shop-floor operations satisfying both temporal and resource constraints [3]. Additionally, criteria such as the throughput, tardiness and the production costs are also considered when optimizing a production schedule. In general, scheduling in manufacturing can be demarcated as an optimization process where limited available resources are optimally allocated over time among operations that are executed simultaneously or sequentially.

When it comes to the main identified issues related to this research, they can be discussed in two different categories; functional and technological. Functional aspect deliberates on the production scheduling process related problems and the technological aspect concentrates on the Multi-Agent approach related opportunities and challenges that have been identified.

1.4. Problem in Brief

Following mentioned are the research challenges that will be considered in this research:

1.4.1. Functional Issues

1) Highly combinatorial aspect of scheduling, which makes traditional approaches that utilize simplified models and centralized computation mechanisms, quite inefficient and ineffective.
2) Real-time dynamic rescheduling and optimization of generated schedules, which is quite harder to implement using traditional approaches due to the large number of constraints that should be considered.

1.4.2. Technological Issues/Opportunities

1) Integrating real-time information of various occurrences when using Multi-Agent approach for a highly-dynamic and adaptive scheduling system.
2) Making the chained dynamic scheduling process more efficient by reducing the time taken for it.
3) Increasing the transparency of the system when performing the dynamic scheduling process by providing feedbacks to the user.

1.5. Proposed Solution

In order to address the challenges and the opportunities mentioned in the previous chapter, the solution proposed in this study is to implement a Multi-Agent based Real-time scheduling system (henceforth referred to as *DynoSchedule*) for the constraint-based scheduling problem in manufacturing. The DynoSchedule is designed to be used by people such as Shop-floor Supervisors, Shop-floor Labours and Production Planners and the main inputs for the system are the order details such as the quantity, required date, priority, scheduling direction and operation details such as the work centre type, materials used. These details are used to dynamically create different kinds of agents as well as to drive the communication between them. The main outputs of the DynoSchedule system are the estimations on the feasible manufacturing window for a set of production orders and the approximation of start and finish dates of a specific order and in turn, operations.

1.6. Resource Requirements

Following are the resource requirements that would be necessary for continuing work on this project.

1.6.1. Computer Hardware Requirements

- PC/Laptop with Intel i5 or i7 Processor, minimum 8GB of RAM
- Printer; for documents etc.

1.6.2. Software Requirements

- Software is expected to run on platforms above Microsoft Windows 7
- Microsoft Visual Studio or NetBeans IDE/Text Editors

- Appropriate Database Servers (Microsoft SQL/MySQL/Object DBs)
- Microsoft Office; to produce documentations, presentations and charts etc.

1.6.3. Other Requirements

- Comprehensive functional knowledge related to Constraint-Based Scheduling in manufacturing.

It should be mentioned that the above-mentioned resources and any additional resource can be acquired by the Author himself, while expecting the required supervision and the guidance from the supervisor and the department in general.

1.7. Structure of the Thesis

The structure of the thesis is as follows: Initially the background and the proposed solution of the study will be briefly discussed in the Introduction chapter. Afterwards, the literature review chapter will discuss the history, the state of the art as well as the future trends of Multi Agent Systems, and its application on dynamic constraint-based scheduling in manufacturing. The technology chapter will elaborate on the different technologies identified and utilized in the proposed system and the Approach chapter will provide an overall image of the proposed solution for the dynamic constraint-based scheduling system.

The Design and Implementation chapters will go into further details on how the system has been designed and implemented using the proper architecture and technologies according to the approach discussed in the Approach chapter. Furthermore, the Evaluation chapter will elaborate on the performance of the system compared to a dataset that has been acquired from a conventional scheduling system that uses both manual and dynamic processes for dynamic scheduling and the Conclusion chapter provides the final verdict on the success of the project and provides its advantages as well as the technical issues and opportunities.

1.8. Summary

In this chapter, the research study that is discussed in this report has been introduced by illustrating its background and motivation, the problem that is addressed in brief, the proposed solution and the resources that is necessary for the completion of this study. Finally, the overall structure of the thesis is also discussed. In the next chapter, the literature associated with the Multi-Agent technology and Multi-Agent based scheduling systems will be discussed and subsequently the identified functional issues and technologies will be discoursed.

Chapter 2

Dynamic Scheduling in Manufacturing – Developments and Challenges

2.1. Introduction

This chapter will focus on the literature associated with the Multi-Agent technology and how it has been used to solve scheduling problems on different domains. Initially a brief history of Multi Agent technology, which is the main technology used in the DynoSchedule system, will be discussed. Afterwards, a critical review of various studies that have been done with regards to Scheduling using Multi-Agent technology in various domains will be discussed. Finally, the functional issues as well as the technical issues and the opportunities that will be identified and addressed in this study will be discussed.

2.2. History of Multi Agent Technology

In the early 90s, pioneers in the domain of Artificial Intelligence such as McCarthy and Nilson have expressed their discontent on existing technologies for AI, indicating that it's important to distinguish between intelligent programs and the tools that they use, aka, special performance systems [4]. While it's important to build new tools, working on tools alone is not going to help the community move towards the original goal of AI; which is to make systems capable of flexible, correct and autonomous actions in different unpredictable and dynamic domains. This is precisely what's enabled by Multi Agent Technology, and according to Russell and Norvig, "AI is the study of Agents" [5].

According to Jennings et al, Multi Agent Systems and Autonomous Agents provides a novel method of studying, designing, and implementing complex software systems. The agent-based perspective provides a repertoire of tools, methods, and metaphors that have the capacity to significantly enhance the way in which people conceptualise and implement various types of software [6]. Multi Agent technology is used in many different types of applications from small scale email filters to large scale mission critical applications such as air-traffic controls. Even though it might appear that this range of application will have very little in common, in contrary, in both these applications the Agent is used as the key abstraction. This is a concept that allows quite natural and easy conceptualization of a wide variety of applications in terms of agents, which makes researchers and developers in the domain to be quite enthusiastic about the potential of this technology in the future.

The interest that has developed about the Multi Agent Technology didn't emerge out of the blue, rather, researchers and developers from many different domains have been closely studying and discussing the capacity of this technology for quite some time. The main contributors for this are:

- Artificial Intelligence
- OOP (Object Oriented Programming) and Concurrent object-based Systems
- HCI (Human Computer Interface) Design

Undoubtedly, one of the main contributors to the field of Multi Agent Systems is the domain of Artificial Intelligence. This is a domain which studies intelligent artefacts, and if such artefacts have the capability of sensing and acting in a given environment, they can be considered as Intelligent Agents [6]. Even though, the Agent technology is one of the main areas in AI, until the 1980s, there was only very little interest and effort within the AI community to study intelligent agents. The main reason for this can be identified as the tendency of AI researchers to study the intelligent behaviour of individual *components* such as learning, reasoning and problem solving. The expectation at the time was that, studying such components independently would prove more successful, and creating intelligent agents by synthesizing these individual components would be quite straightforward. This assumption seems to have been implicit within the AI researchers in the throughout 1970s. However, one exception to this rule was the area of AI planning, on which there were researches connected to intelligent agents.

Early AI planning research that was conducted during the 1970s and early 1980s focussed primarily on the different representations required for actions, the planning algorithms and the efficiency of them. Although, some micro-world examples seemed to provide reasonable performance, it was quickly observed that they do not scale well with large realistic scenarios. This apparent failure in early researches on AI planning techniques to scale to cater the real-world scenarios, led many researchers during the mid-1980s to discuss and question the viability of conventional reasoning approaches for AI and to explore different methodologies to solve these issues. One of the best-known such critics can be named as Rodney Brooks, who presented various different objections towards symbolic AI modelling through a series of published papers. As a part of his research, Brooks developed an agent control architecture, known as the "*Subsumption Architecture*", which did not employ any sort of symbolic

reasoning or representations; rather utilized a collection of intelligent agents with task accomplishing behaviours.

By the early 1990s, many researchers identified that reactive architectures can be only applied to certain domains and problems while being less suitable for others, in fact, most problems didn't work with purely reactive or deliberative architectures. This led many researchers to investigate on hybrid architectures, which tried to synthesize the best aspects of both reactive and deliberative approaches. In these hybrid architectures, the Subsumption Architecture, implemented by a collection of agents with task accomplishing behaviours, played an important role.

Later on, Practical Reasoning Agents started to develop in the area of agent architectures, which took inspiration by a theory of practical reasoning, or the pragmatic reasoning, done by humans. This has long been an area of study in different domains, especially in Philosophy, who developed various theories that can account for human behaviours. These theories influenced many researches in practical reasoning architectures, among which, the best-known type of architecture is known as the Belief-Desire-Intention (BDI) model. As the name implies, the agents in this architecture are characterized by the different states of mind; beliefs, desires and intentions. Beliefs refer to the information a particular agent possesses about its environment, Desires are known as the different options or course of actions available to the agents, while Intentions correspond to the state of affairs that are already committed by the agent.

The philosophical foundations of the Belief-Desire-Intention model can be found in Bratman's perspective of how the Intentions influence the practical reasoning system of humans. There are various different BDI agent systems that have been implemented, Procedural Reasoning System (PRS) probably being the best known. There are different logical theories of BDI systems developed by researchers who are interested in practical reasoning architectures. One of such theories is Shoham's proposal for "agent-oriented programming, a multi-agent programming model in which agents are explicitly programmed in terms of mentalistic notions such as belief and desire" [6].

2.3. Application of Multi Agent Technology in Scheduling

A constraint-based scheduling problem decides on when to carry out different shop-floor operations satisfying both temporal and resource constraints [3]. Additionally, criteria such as the throughput, tardiness and the production costs are also considered when optimizing a production schedule. In general, scheduling in manufacturing can be demarcated as an optimization process where limited available resources are optimally allocated over time among operations that are executed simultaneously or sequentially.

When it comes to the main identified issues related to this research, they can be discussed in two different categories; functional and technological. Functional aspect deliberates on the production scheduling process related problems and the technological aspect concentrates on the Multi-Agent approach related opportunities and challenges that have been identified.

2.4. Functional Issues

Scheduling problems exist in various domains such as manufacturing organizations, educational institutes, transportation companies and medical institutes, and is typically considered NP-Hard (Non-Deterministic Polynomial Hardness), which basically indicates that the computation time exponentially increases with the problem size. However, production scheduling is considered one of the hardest scheduling problems not only because it typically deals with a large number of constraints such as resources, work centres or machineries, tools, skilled personnel and suppliers, but also due to its highly dynamic and uncertain nature [7].

One classical scheduling example is where a set of production orders and machines are given, and each order comprises of a list of operations that should be processed without any interruption on a specified machine for a given period of time, and the goal is to find the most optimal schedule to allocate these operations on the available machines.

1. The total number of possible solutions for a production scheduling issue of this nature, with n jobs and m machines is $(n!)^m$ [7]
2. If the skilled operators who operate the machines are also considered, the total number of possible solutions for a production scheduling problem with n jobs, m machines and k operators could be $((n!)^m)^k$

In cases where production planning and scheduling are done at the same time, traditional approaches consider these processes separately and often sequentially, resulting in suboptimal solutions. Integrating these processes can theoretically result in a highly optimal global solution, however, it will also significantly increase the solution space.

The most prominent issue when it comes to scheduling in industrial applications of any domain, varying from transportation [8] to manufacturing, is handling unforeseen dynamic situations and optimizing the schedule accordingly. Especially when it comes to manufacturing, things rarely go as initially planned since there can be various unanticipated events like machine breakdowns, lack of skilled operators, shortage of materials, delayed supplies etc. With the conventional approaches, these sorts of real-time events do not reflect on the already generated schedule, often resulting in tardy or delayed orders, inefficient usage of resources and overall customer dissatisfaction.

2.5. Technical Issues & Opportunities

Multi-Agent approach for scheduling problems has become quite popular in the past few years and various research have been conducted on different types of scheduling domains on how to address various scheduling problems using agent technology. These applications include transportation scheduling [9], logistics [10], production scheduling [11] [12] [13] [14] [2], computer job scheduling [15] etc.

When going through the literature on Multi-Agent based approaches for scheduling, there are various challenges that can be identified.

1. Identifying proper negotiation mechanisms, frameworks and protocols for agents. Since most current MA approaches utilize quite basic negotiations [16] or bidding techniques for agents, more powerful negotiation mechanisms and protocols are required to be investigated upon.
2. Integration of real-time information for a more dynamic and adaptive scheduling system. Given the highly uncertain nature of manufacturing processes, the generated schedules can easily be outdated if they are not optimized utilizing real-time information. Therefore, real-time dynamic rescheduling can be named as one of the most important research issues.

3. Integrating Multi-Agent based approach with other technologies for a superior solution. While the multi-agent approach allows more flexibility and adaptability and is more suitable for dynamic rescheduling, traditional search methods such as Genetic Algorithms or Simulated Annealing can be used to focus on more optimal scheduling solutions. Therefore, combining these technologies would theoretically result in more powerful production scheduling systems.
4. Identifying benchmarks for evaluating Agent-based scheduling systems [17]. This has come up as an important research issue, since it is important to compare and contrast between different Multi-Agent based approaches as well as traditional approaches with the multi-agent approach for production scheduling.

Over the past decade, Multi-Agent approach has emanated as one of the most prominent technologies that can be used for different types of scheduling problems. George Rzevski et al. conducted various researches on Multi-Agent based Real-time scheduling system for Taxi companies [9] as well as rent-a-car companies [8]. The main issue they are aiming to address with these researches is the management of complex transportation networks in real-time, which has been quite challenging to be addressed by the prevailing software systems. In addition, the existing solutions also don't deal with latest technologies such as the GPS, GLONASS navigation, Electronic Maps, or the latest services that provide updates on traffic and weather conditions in different areas etc. One of the main reasons for this is that such solutions are generally highly dependent on PC or other stationary devices rather than mobile devices.

As a result of these issues, scheduling tasks are usually delegated to highly qualified and experienced dispatchers by the transportation companies, implementing only a limited set of features in existing ERP systems. In order to resolve such issues, these studies have proposed different Multi-Agent approaches such as based Adaptive Scheduling Systems [9] and Demand-Resource Networks (DRN) [8] conception.

The main task considered in a rent-a-car domain is "delivering a car to a customer", and the scheduling process starts by distributing this into several subtasks. Each of these sub-tasks are then scheduled independently by initiating a complex agent negotiation mechanism. For instance, generally a car is washed before it's delivered to a customer; in such a scenario, at the beginning of the aforementioned task, an agent who's responsible for a "station pickup" subtask, identifies the best available car that can be picked up at a station where all the vehicles

are parked, and subsequently creates a “wash the car” subtask. The agent who’s responsible for that subtask then identifies a driver who’s capable of carrying out that particular task. Per se, in rather complex scenarios, there can be quite a large number of subtasks that are generated in relationships of cause and effect by matching the demands with resources, to fulfil the relevant tasks.

In addition to demand-resource driven interactions, the agents also have the capability of identifying conflicts and resolving them [8]. For example, in case if an agent tries to reserve a resource that has already been allocated for another tasks, the agents have the capacity of resolving such conflicts through negotiations. Moreover, the interactions between the agents are not static, rather they are created or broken dynamically based on various different criteria related to the tasks that a particular agent needs to perform. This helps the system to not only create schedules but also to proactively improve the schedules when an unforeseen event occurs.

Following listed are some of the advantages of the system discussed in this study provides over the conventional scheduling solutions:

- Agent based approach allows modelling the high complexity of modern business by using new agents that can be used to represent different concepts such as resources tasks or interest;
- The system provides a great solution for the optimization problems with large number of factors that are highly dynamic and interrelated.
- Allows the system to do dynamic real-time scheduling through resolving conflicts of the existing schedule by taking into account the events that occur
- The system has the ability to explain the decisions taken by it, which allows users to understand the schedule better and to take corrective actions if there are any issues.
- The system provides a high level of flexibility by allowing the business to model various criteria that need to be considered when performing the scheduling tasks.

The developed system in this study can be named as a one of the first industrial multi-agent systems for real-time dynamic scheduling. It demonstrates highly intelligent behaviour such as the capability of perceiving real-time events and generating schedules making the maximum utilization of available resources and monitoring the schedules and proactively improve it when required. It also strives to improve the specified KPIs by self-regulating its own activities.

Another similar study was done by Santos & Madureira [2], who proposed a Multi-Agent System for Dynamic Manufacturing Scheduling using Meta-Heuristics (MADynScheMH) for manufacturing organizations. It proposes the concepts of Meta-Heuristics and Multi-Agent approach in combination to solve the dynamic scheduling problem in manufacturing. According to their proposal, the agents of the system represent different entities such as machines, work centres and operators, with the objective of scheduling their problems locally and unifying each of those local schedules to create a single global solution.

In addition, this study proposes the use of Meta-Heuristic methods such as Tabu Search and Genetic Algorithms to deal with the dynamic nature of the domain, which has the ability to adapt by reusing or changing the solutions/populations depending on the dynamism of the problem. It suggests that that self-parameterization of such Meta-Heuristics will allow better adaptability for different situations that are considered.

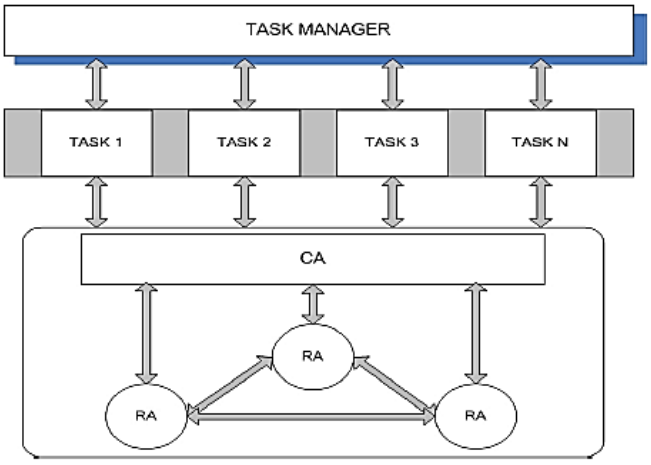


Figure 1.Proposed Multi-Agent Architecture of the MADynScheMH System

The proposed model utilizes a Repair Approach (RA), which as the ability to initially wait for the solutions provided by the local resource agents and then apply an optimization (repair) mechanism to shift operations generated till a globally feasible solution is obtained. This method allows for better optimization of all the integrated schedules generated locally by the resource agents.

The Coordination Approach (CA) has been used to provide dynamism to the agents by providing new jobs, cancelling existing jobs or changing different parameters of the job. These two approaches (Repair and Coordination) in combination are used in order to guarantee the feasibility of the generated schedule, since the integration of locally generated schedules will not always be feasible to carry out the tasks.

A similar study was done on Reactive Agent mechanisms for scheduling the chemical Processes when manufacturing by Demazeau et al [18]. There are various types of chemical processes carried out in chemical baths that the production items, especially electrical components, should undergo during a manufacturing process; this is known as the anodization process. The recipe only indicates the minimum and the maximum amount of times for each bath, and it depends heavily on the different characteristics of items. In this research, the PACO paradigm, which is a contraction of coordinated patterns, has been applied for the reactive agents to develop the solution. There, the researches have applied an interaction mechanism similar to physical forces, which allows the agents to fulfil personal as well as global constraints.

The manufacturing system consists of about 50 baths, typically even though a given recipe does not have to go through all the 50 baths, it will still have about 15-25 baths that should be visited, and there's room available to overcome bottlenecks by properly scheduling the additional baths of same type, since the processing time highly depends on the type of the bath. There are 3 slightly overlapping cranes that has the ability to move bars from one location to another within an array of bars. In addition to these, there's another important component in the system, which is the input buffer, which consists of about 30 bars that are available to be processed. Figure 2 gives an overview of the production system:

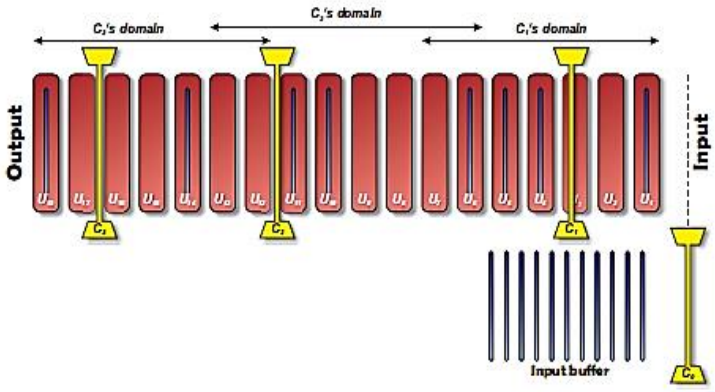


Figure 2.Overview of the Bathing Process

In the developed solution for scheduling the above system, the researchers have applied the PACO paradigm for agents, which indicate that the agents are purely reactive, thus they do not have specific internal representations about themselves, other agents or the environment, rather they only respond to various events happening in the environment. This paradigm can be defined by three fields, by dividing the agent model:

- **Perception field:** used to determine the agent's perspective about its environment.
- **Communication field:** used to determine the agents who have the ability to interact with each other
- **Action field:** is used to determine the area where agents can perform different actions.

Here, the agents represent each step (one bar) of the recipe, and agents that belongs to a specific bar forms a group. An agent has the initial knowledge about the type of the bath a particular bar should go to and the minimum and the maximum times it should remain in the bath, as well as the information about the previous and the subsequent agents with respect to the recipe step. However, a given agent doesn't have any idea about rest of the agents or any other aspects of the system, nor do they have the ability to communicate with the rest of them. In order to succeed, a particular agent must visit the appropriate bath for the right amount of time specified by the recipe.

In the developed system, each agent has a specific set of goals to be accomplished as below:

- Go to the relevant bath
- Stay closer to the previous agent of the group

In addition, agents also have a set of constraints:

- Maintaining distance to the minimum and maximum time required
- Assist the subsequent agent to remain closer
- Assis the rest of the agents to fulfil their goals.

These targets and constraints allow the agents to maintain proper interaction and communication between each other while fulfilling their own goals. When there are multiple agents from different groups that share the same time slot for a given bath, it has to be negotiated between themselves.

In order to validate the PACO paradigm used for the agents, the researchers have carried out various tests to see if the system has the ability to create valid plans for the bars, comparing the satisfactory rate for agent groups with different number of agents: where 100% represent fully satisfactory plan, which indicates that for a specific bar, all the visits to baths in the recipe complies with the appropriate minimum and maximum time frames. Following displayed is the satisfaction rate chart for different agent groups:

As visible in figure 3, the 100% satisfactory plans are created in the 1 group scenario, and the 5-group scenario fluctuates around 90% satisfactory level. The other agent groups are slightly lower and should be improved. However, this indicates that the proposed solution has the ability of creating efficient plans.

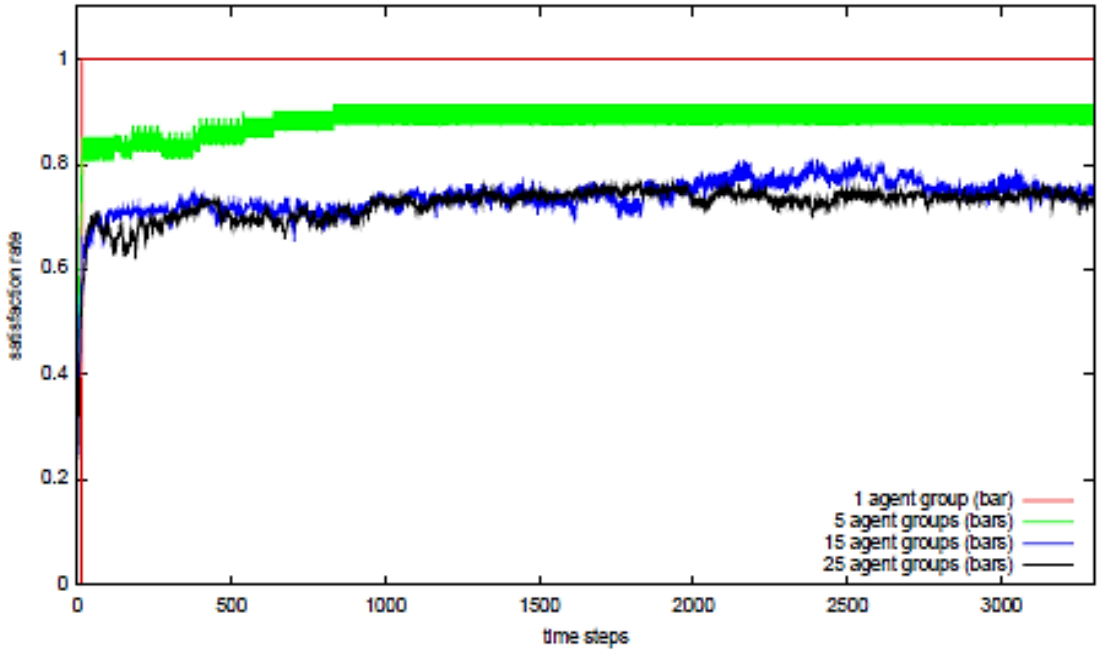


Figure 3.Satisfaction rate with 1,5,15 and 25 agent groups. Source: [11]

Another research related to the Integration of Process Planning and Scheduling (IPPS) was conducted by Li, X et al [19], which is a process that is employed in manufacturing organizations, where, provided with an *n* no. of parts to be manufactured on machines by different operations and resources, select the most appropriate resources and the order of operations that most fits the production of that part in order to create a schedule which will satisfy the appropriate constraints and achieve the objectives.

In the research the authors have suggested an IPPS framework which is based on Multi-Agent technology, which employs 3 main types of Agents and multiple databases. These agents are known as:

- *Job Agent*
    - Represent a job. Contains information regarding the particular job such as the Job ID, type, due dates, CAD Drawings, quantities, tolerance, quality requirements, finishing of the surface etc. These agents also utilize multiple

status such as idle and manufacturing operation. These agents acquire information from the database and negotiate directly with machine agents in order to generate the schedule.

- *Machine Agent*
    - Represent different resources/machines. These agents include information such as the machine ID, manufacturing features that a machine can process, the time that it will be taken for the manufacturing process, and the total amount of time taken for the process etc. These agents also utilize statuses such as idle, manufacturing operation, breakdown etc.
- *Optimization Agent*
    - Optimization is the most important type of agent available in the research, which is assigned with the task of optimizing the process plans and scheduling plans to find an effective solution. In order do this optimization process, the Optimization agent utilizes an Evolutionary algorithm.

Figure 4 illustrates the main structure of the Agents available in this study:

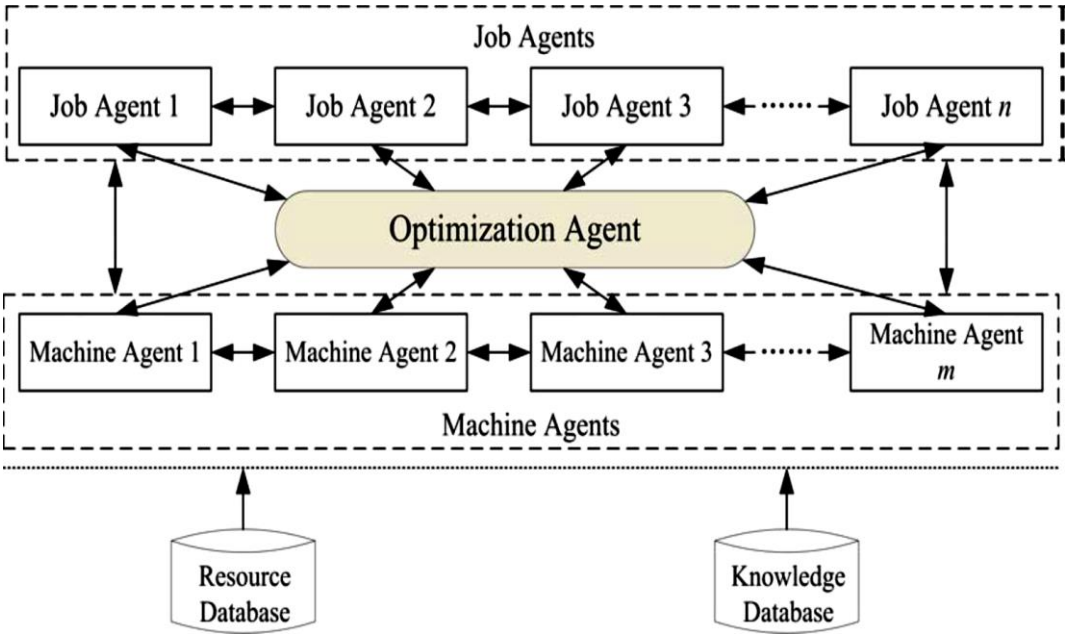


Figure 4.Agent architecture for the IPPS system

This system has been developed in a simulated manufacturing system using Microsoft Visual C++ programming language as the core of its framework. Microsoft Access database has been used to store and view information related to resources and jobs.

The agents of the system are executed on three different host machines and the inter-agent communication has been based on a peer to peer communication method based on the TCP/IP network protocol. In addition, the communication is managed by KQML or Knowledge Query Manipulation Language, hence all the messages passed between the agents are compliant with the KQML's standard performatives set.

Figure 5 displays the implementation of the system on a simulated manufacturing organization system:

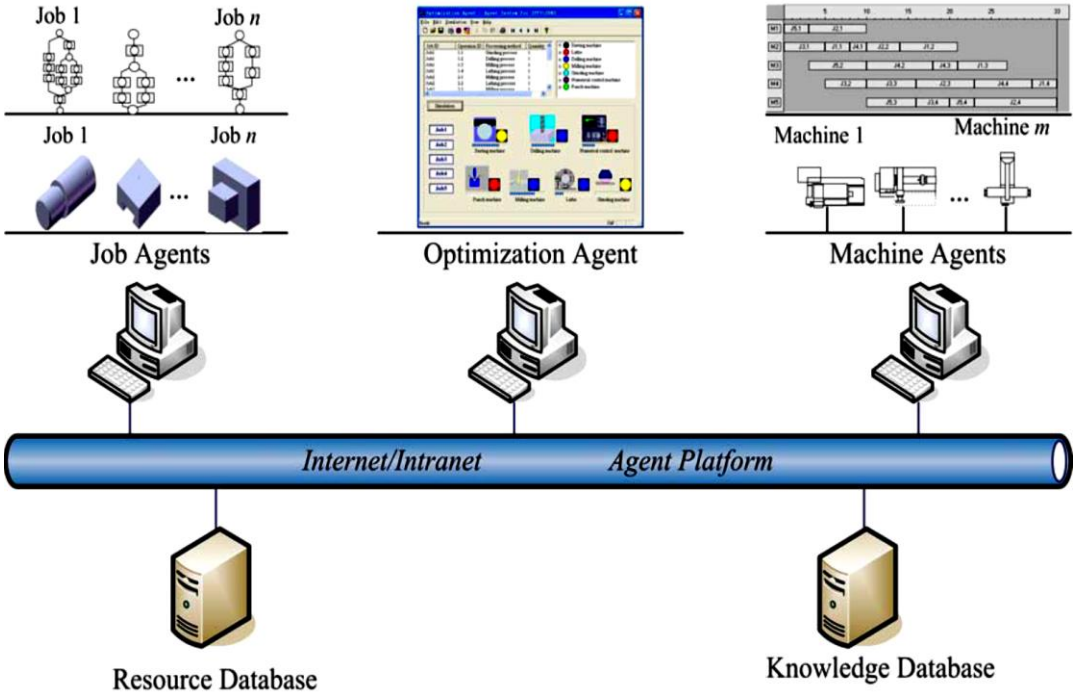


Figure 5. Manufacturing Simulated System

After the scheduling requirements and the availability of the resource are provided to the system, the Job Agents, Machine Agents and the Optimization agents will communicate, coordinate and negotiate in order to find and optimize alternative process plans and scheduling plans.

When it comes to the experimental results of the system, the authors of the study have considered the methods suggested by Sundaram and Fu [20], by initially considering five jobs and five machines and later by extending the method to considering a higher number of jobs and machines. When considering the results of the experiments, the authors have concluded that the proposed method can identify better scheduling plans and has the ability to get fast evolution speeds on a reasonably high search space.

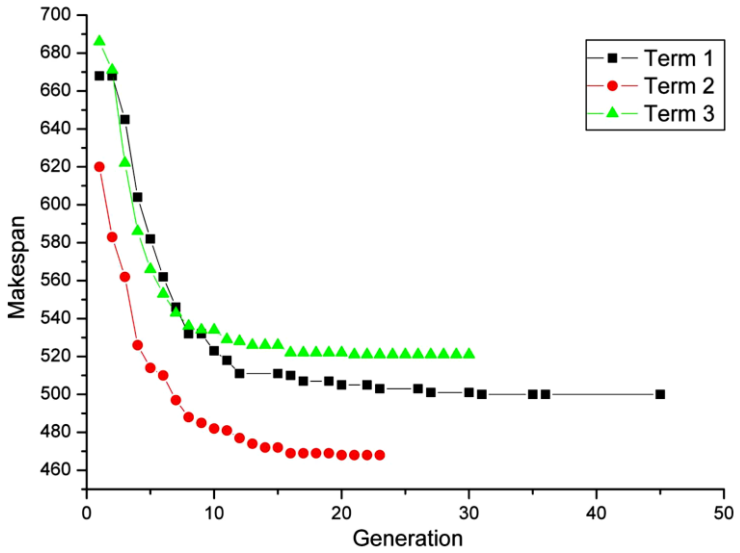


Figure 6.Convergence curves of the system in three terms

Figure 6 displays the convergence curves of the implemented system when comparing three terms. Finally, the authors conclude that the system can be utilized as an acceptable approach for IPPS due to multiple factors:

- The approach considers all the necessary parameters for manufacturing process planning and scheduling.
- It obtains better results than the existing approaches.

One of the more recent studies done on this domain is the use of Hyper-Heuristics based Framework for the production scheduling problem using Multi-Agent technology [21]. In most real-world combinatorial issues, heuristics approaches are used often even though they don't quite promise an ideal solution, rather an "acceptable" solution in a reasonable amount of time. According to the study, hyper-heuristic approach is a method that tries to automate the procedure of identifying, coalescing, generating or adapting multiple simpler heuristic methods in order to solve computational search issues more efficiently. Essentially, hyper-heuristic concept comprises of multiple-levels of heuristics where higher-level and lower-level heuristics coordinate with each other. This is a more dynamic approach that can be applied for issues that are quite large in scope, by searching in the heuristics space rather than the solution space directly. Figure 7 shows a general framework for the hyper-heuristic approach:

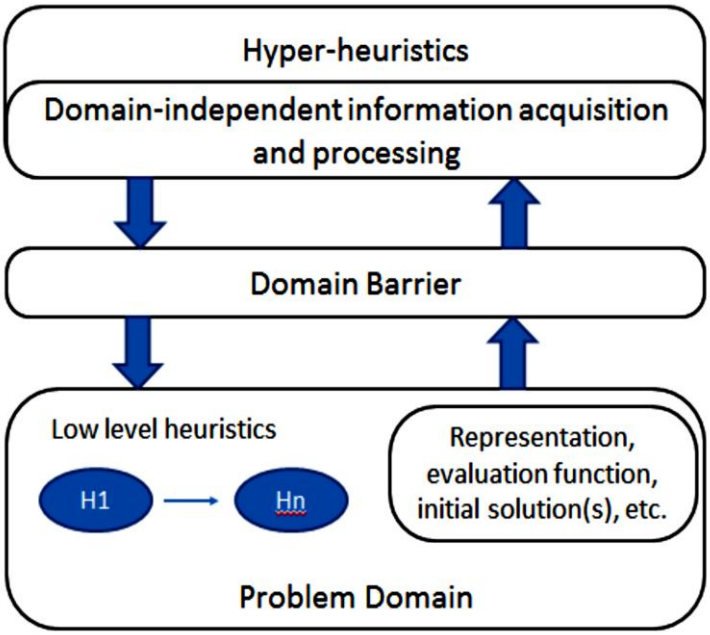


Figure 7. Hyper-heuristic approach framework

In the proposed study, the authors have introduced 9 different types of agents along with the hyper-heuristic approach to solve the problem of production scheduling. Following listed are those 9 agents:

- *Problem Agent*: Responsible for initializing other agents and coordinating with the Trainer Agent about the problem description.
- *Trainer Agent*: acquire the problem details from the Problem Agent and train the system using different training sets.
- *Training Dataset Agent*: allocated with the task of managing the training dataset and providing it to Algorithm Agents.
- *Heuristic Pool Agent*: manages the heuristics collection.
- *Algorithm Agents*
    - *Genetic-Algorithm Hyper Heuristics*
    - *Simulated-Annealing Hyper Heuristics*
    - *Genetic-Programming Hyper Heuristics*
        - The main purpose of these 3 types of agents are to run the hyper-heuristics algorithm when parameters and heuristics are received and send the best solution to the optimizer agent.
- *Advisor Agent:* advise about the most suitable heuristic for a given problem to the Solver Agent.

- *Solver Agent:* tries to solve the problem received by the problem agent with the most appropriate heuristic received from the Advisor Agent.

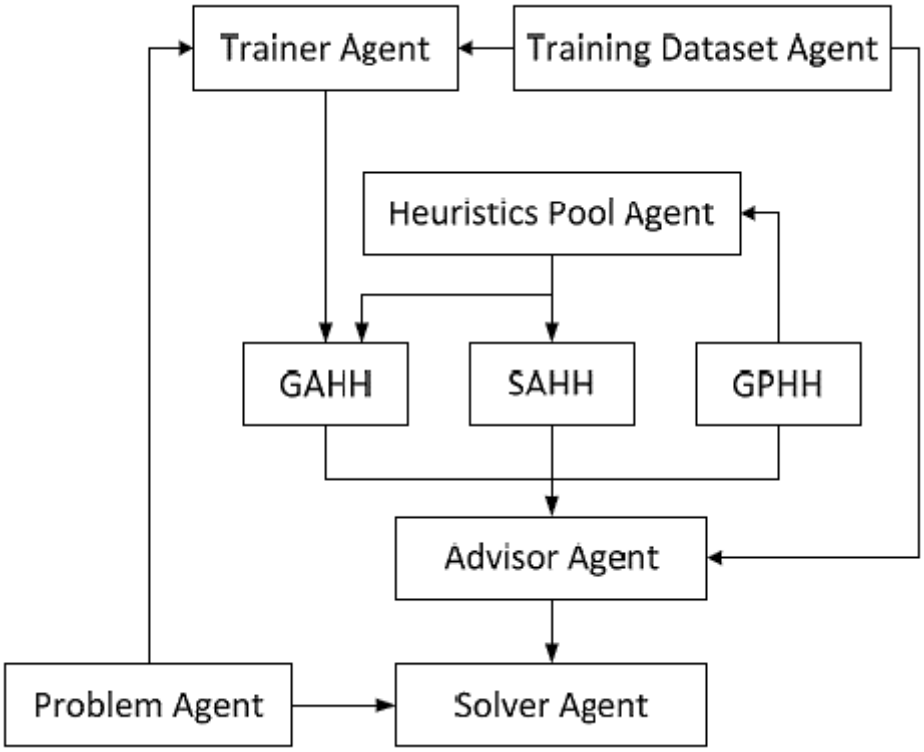


Figure 8.System Architecture

The proposed system has been implemented using a computer application with some experiments conducted to identify its performance. There, it has used a single machine for job scheduling and the average performance are compared for 3 different algorithm agents and results were obtained. There, the authors have identified that the results display that when using GPHH algorithm, it provides much better results when compared to using GAHH, and SAHH as shown in the figure 9:

ID	HEURISTIC	MACHINE 1	MACHINE 2	MACHINE 3
H1	MRT	9.12	10.27	151.8
H2	SPT	86.63	26.55	117.72
H3	LPT	136.15	77.16	315.98
H4	EDD	12.86	9.94	153.64
H5	LDD	137.76	63.02	241.07
H6	MON	86.63	26.55	117.72
H7	GPHH	**8.49**	**9.94**	**114.79**
H8	GAHH	68.62	36.9	200.64
H9	SAHH	99.01	50.08	246.73

Figure 9.Performance of 6 benchmark heuristics with GPHH, GAHH, and SAHH

The approach proposed in this study tries to be more generic when compared to other studies that have been discussed previously.

2.6. Summary

In this chapter, it was deliberated on the various topics related to the technologies and literature of this research domain. Initially, the history of Multi Agent Systems was discussed in detail, and subsequently, the application of MA technology in dynamic constraint-based scheduling was detailed. Later, the current challenges and opportunities related to the dynamic constraint-based scheduling systems were investigated. Following table 1 summarizes the discussed studies in the literature review:

Research	**Summary**	**Reference**
Multi-Agent Real Time Scheduling System for Taxi Companies	Introduced Adaptive Scheduling Systems conception	G. Rzevski, P. Skobelev, A. Ivaschenko, and A. Glaschenko, "Multi-Agent Real Time Scheduling System for Taxi Companies," *Proc 8th Int Conf* ….
A Multi-agent Scheduler for Rent-a-Car Companies	Using Demand-Resource Networks (DRN) Conception	S. Andreev, G. Rzevski, P. Shviekin, P. Skobelev, and I. Yankov, "A Multi-agent Scheduler for Rent-a-Car Companies," *Lect. Notes Comput. Sci.*, pp. 305–314
Multi-Agent System for Dynamic Manufacturing Scheduling using Meta-Heuristics	Combining the concepts of Meta-Heuristics and Multi-Agent approach to solve dynamic scheduling	J. Santos and A. Madureira, "Proposal of Multi-Agent based Model for Dynamic Scheduling in Manufacturing."
Reactive agent mechanisms for scheduling manufacturing processes	Reactive Agents with PACO paradigm	Y. Demazeau, K. Hallenborg, and A. J Jensen, "Reactive agent mechanisms for scheduling manufacturing processes."

Multi-agent-based integration of process planning and scheduling	Using Agents with Optimization techniques (Evolutionary algorithm)	X. Li, W. Li, L. Gao, C. Zhang, and X. Shao, "Multi-agent based integration of process planning and scheduling," in *2009 13th International Conference on Computer Supported Cooperative Work in Design*, Santiago, Chile, 2009, pp. 215–220.
Multi Agent Hyper-Heuristics based framework for production scheduling problem	Higher number of agents with a Hyper-Heuristic approach to solve the production scheduling issue	C. E. Nugraheni and L. Abednego, "Multi Agent Hyper-Heuristics based framework for production scheduling problem," in *2016 International Conference on Informatics and Computing (ICIC)*, 2016, pp. 309–313.

Table 1.Summarization of the Literature review

In the next chapter, the Technology that is used in this system will be discussed in greater detail.

Chapter 3

Technology

3.1. Introduction

This chapter will focus on the main technology that is used in the DynoSchedule system, which is Multi-Agent technology, and go through the fundamental concepts that are used in Multi Agent technology such as Situatedness, Autonomy and flexibility among others. In addition, this chapter will also discuss about the framework used to develop the Multi-Agent component of the DynoSchedule system, and also go into further details regarding the other technologies used in the development of it such as the programming language, database technology and the version controlling system.

3.2. Multi Agent Technology

The main technology used in the proposed system is Multi Agent Technology; therefore, it's crucial to properly understand the technology before moving on to the subsequent chapters of this study. First, it's important to define what is meant by such terms as "agent", "agent-based system" and "multi-agent system". Unfortunately, this is not quite easy, as some key concepts in the field lack universally accepted definitions. Even up to this date, there's a lot of deliberation going on regarding what exactly is meant by an Agent. Even though this might be considered as an obstacle to its development, the AI community still has been able achieve leaps and bounds when it comes to agent-based technology, without such universally accepted definitions. Nevertheless, it is worth spending some time on the issue, otherwise the terms that is used in the remainder of this study will come to lose all meaning. Therefore, an agent can be defined as a computer system, situated in some environment that is capable of flexible autonomous action in order to meet its design objective. There are thus three key concepts in our definition: situatedness, autonomy, and flexibility [6].

Situatedness, in this context, means that the agent receives sensory input from its environment and that it can perform actions which change the environment in some way. The physical world wide web can be named as an example for where Agents are situated in different locations, which is in contrast with concepts such as Expert Systems that discusses disembodied intelligence. As an example, MYCIN, which is a paradigm expert system, did not have any interactions with the physical environment. Rather it received information through a middle

man via different sensors [6]. Similarly, it did not act upon the environment, rather it provided feedback to different third parties. The second key concept in Agent Technology is Autonomy which is again hard to be defined precisely, however, in layman terms, it means the ability to work on its own accord without human intervention, and the ability to control its actions and the state. In a more defined sense, Autonomy can also refer to systems that can learn and adapt on its own.

There are various examples for situated and autonomous systems at present including: any process control system, which must monitor a real-world environment and perform actions in order to modify its conditions automatically; these systems can vary from simple thermostats to quite complex and sophisticated nuclear control systems. Another similar example would be software daemons, that has the ability to monitor software environments and modify its conditions by performing various actions; the UNIX xbiff program can be named as an example for such a system, which monitors a user's incoming email and obtains their attention by displaying an icon when a new, incoming email is detected.

While the above are certainly examples of situated, autonomous systems, they cannot be considered as agents since they are not capable of flexible action in order to meet their design objectives. By being flexible, it implies that the system is;

- **Responsive**: Agents should perceive their environment and respond in a timely fashion to changes that occur in it.

- **Pro-active**: Agents should not simply act in response to their environment, they should be able to exhibit opportunistic, goal-directed behaviour and take the initiative where appropriate.

- **Social:** Agents should be able to interact, when appropriate, with other artificial agents and humans to complete their own problem solving and to help others with their activities.

However, naturally, some agents will have additional characteristics, and for certain types of applications, some attributes will be more important than others. However, the common belief is that it is the presence of all the attributes in a single software entity that provides the power of the agent paradigm and having such attributes differentiate agent systems from software paradigms such as the OOP-based systems, expert systems and distributed computing systems.

With the basic building block notion of an agent in place, more related terminology can be defined. By an agent-based system, the key abstraction used is that of an agent. In theory, an agent system can be designed by conceptualising different types of agents and implemented without any software structure that represent them. This can be identified in parallel with object-oriented software, where it is entirely possible to design a system in terms of objects, but to implement it without the use of an object-oriented software environment. However, this can be quite counter-productive and unusual. A similar situation exists with agent technology; therefore, it is expected that an agent-based system to be both designed and implemented in terms of agents.

As defined previously, an agent-based system may contain one or more agents, although there are circumstances where a single agent would also suffice. A good example is the class of systems known as expert assistants, wherein an agent acts as an expert assistant to a user attempting to use a computer to carry out some task. However, when a system is designed in a pure Multi-Agent structure using different types of agents incorporating different features, it can be quite interesting and sophisticated in terms of a Software Engineering perspective. Multi-Agent systems are best used in order to solve issues that have different methods of solving them, different standpoints and/or even different types of entities with the ability to solve that problem. Typically, such systems have the advantage of more sophisticated interaction patterns. These interaction patterns include cooperation (working towards a common goal together); coordination (proper organization of an activity that solves a problem while minimizing damaging interactions and exploiting advantageous interactions); and negotiation (all the involved parties coming to an agreement that is acceptable). It is the flexibility and high-level nature of these interactions which distinguishes multi-agent systems from other forms of software, and which provides the underlying power of the paradigm.

3.3. Popular Frameworks for Multi Agent System Development

Nowadays, numerous tools are available for the development of Multi Agent based systems. Such tools have facilities for naming, executing, managing the execution, accessing system resources, maintaining integrity and protection of agents and the platform, as well as for supporting the migration of location and communication services. While there are hundreds of tools available for multi agent systems development [22], in this research, JADE Multi-Agent development environment has been selected.

Among others, JADE is the most generic and the most commonly used Multi-Agent development framework in this domain. JADE comprises of a middleware system that has been built in compliance with the FIPA specifications, which simplifies the implementation of Multi Agent System. In addition, it also provides the developers with a graphical toolset to support easy troubleshooting and deployment [23]. A JADE-based system can be distributed across multiple machines, and the configuration can be controlled via a remote GUI. The configuration can be changed even at run-time by moving agents from one machine to another, as and when required. Apart from the agent abstraction, JADE also comprises of a powerful task composition and execution model, an agent communication model with peer-to-peer support based on the asynchronous communication paradigm, a yellow-pages service that allows for publishing and subscribing mechanisms and various other highly advanced features. Due to these capabilities and its reputation, JADE has been selected as the base system for this research.

Figure 10.JADE Multi Agent Development Framework Logo

In addition, MadKit was also considered as a strong candidate for the development of the system in this research. It's another lightweight, Java-based Multi Agent development framework that also can support other object-oriented programming languages such as C++ through plugins.

3.4. Other Technologies Used in DynoSchedule System

3.4.1. Programming Language

Apart from JADE Multi Agent development framework, there are various other tools and technologies that were used in the DynoSchedule system. The first and the foremost important decision was to decide on a proper programming language to be used for the core system implementation. As such, Java Programming language was selected as the main programming language due to following reasons [24]:

- **Simplicity:** Java language is a common, easy and efficient language to develop applications with, yet it also offers a lot of useful features to the programmers such as automatic memory allocation and garbage collection etc.

- **Platform-independent:** Since Java offers the ability to run a written program on any hardware or a software platform that supports JVM, it allows easy compatibility from different computer systems.

- **Rich API and Library Support:** This is one of the main reasons for selecting Java as the programming language for the implementation of the DynoSchedule system. Since JADE, which is the framework that is selected for Multi Agent development, is mainly supporting Java and due to the larger number of additional APIs and libraries that can be used when developing, Java was the prominent choice of language.

- **Light-weight and Powerful:** Another important deciding factor when selecting Java, is its ability to make the application light-weight and run quite efficiently, which is important for applications such as DynoSchedule that relies heavily not only on Multi-Agent based communication but also on the Business Logics that are relevant for the proper functionality of the system.

By considering the above factors, Java has been selected as the main programming language when developing the DynoSchedule system's Business Logic Layer and the Multi-Agent module.

3.4.2. User Interface

According to the architecture of the DynoSchedule system, the User Interface was decided to be developed on a separate layer that will communicate with the Business Logic layer of the system. As such, it was decided to develop a Web Application as the UI for the system with the use of *JavaScript* and *React,* which is a popular JavaScript library that can be used to develop rich and sophisticated User Interfaces.

React is a library developed and maintained by Facebook Inc. that provides a large amount of features to create Component-Based UIs, where each component has the ability to manage its own state [25].

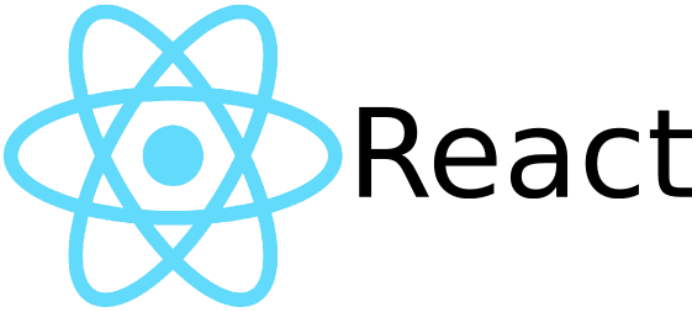


Figure 11.React JavaScript library for UI

3.4.3. Database Technology

Another important module for any Enterprise application is its database technology. Since the DynoSchedule system is aimed at manufacturing organizations that manage their day to day production schedule, it is vital to provide them the ability to manage their data with an appropriate Database Management System. At present, there are numerous DBMSs available both based-on SQL such as MS SQL, Oracle, MySQL and No-SQL such as MongoDB, Hadoop, Cassandra. However, since enterprise level organizations need to keep track and interact with their databases more frequently, MySQL relational database has been selected as the main DBMS for the DynoSchedule system.

Figure 12.MySQL logo

Additionally, Amazon Web Services were used to host the MySQL database in the cloud, so that the system can access those data from anywhere.

Figure 13.AWS

3.4.4. Version Controlling

Finally, for the version controlling of the system, GitHub has been used.

Figure 14.GitHub Logo

The system will be available for reference on the following GitHub repositories:

- **Core: https://github.com/prabash/DynoScheduler**
- **UI: https://github.com/prabash/dyno-scheduler-ui**

Since the core and the User Interface of the DynoSchedule has been developed independent from each other using different technologies (Java and JavaScript), they have been maintained on different GitHub repositories.

3.5. Summary

This chapter mainly deliberated on the Multi Agent technology which is the main technology used in the DynoSchedule system. In addition, other technologies used in the development of the DynoSchedule system were also discussed such as Java as the main programming language for the development of the core, JavaScript based ReactJS as the main development library for the UI and the use of GitHub as the version controller for the system. In the next chapter, it will discuss the approach of the DynoSchedule system.

Chapter 4

A Multi-Agent Based Approach to Dynamic Scheduling in Manufacturing

4.1. Introduction

This chapter discusses regarding the approach that the proposed DynoSchedule system has taken when using the previously discussed Multi Agent technology to develop a Dynamic Scheduling system for the constraint-based scheduling problem in manufacturing organizations. Per se, it will discuss about the main hypothesis, inputs, outputs, process, users and the features of the system in a higher level.

4.2. Hypothesis

The main hypothesis of this project is that implementing a Multi-Agent based Real-time scheduling system, DynoSchedule, would be a solution for the previously discussed issues such as inefficiency due to long dynamic scheduling chains, lack of transparency, feedback etc., in constraint-based scheduling problem of manufacturing organizations, and overcome issues with the traditional machine learning approaches such as requiring large amount of past data, and the time taken to provide the solution:

Following discussed are the identified inputs, outputs and the features of the system:

4.3. Inputs

Following listed are the main basic inputs of the DynoSchedule system:

- Order lot size (Quantity)
- Order need date (Required date)
- Scheduling Direction (Backward/Forward)

4.4. Outputs

Following are the main identified outputs of the DynoSchedule system:

- Estimating the feasible manufacturing window for a set of production orders.
- Approximation of start and finish dates of a specific order.

4.5. Process

The main process of the system includes the following steps:

- Input the details about production orders to be carried out into the system; including information such as the actual need date, earliest need date and the quantity to be manufactured.

- When entering the shop order/s to the system, the user can set the direction on which the shop order should be scheduled; Backward or Forward. Forward scheduling will do the scheduling from the earliest possible date, while Backward Scheduling will schedule the production backwards from required date.

- The proposed system will take over after this and dynamically adjust the already scheduled shop orders and their operations, if necessary, in order to facilitate the current shop order meeting the required date, as well as the others. There are different criteria that will be evaluated when doing this dynamic scheduling process such as:
    - *the orders that have a later required date than the current date, the orders that have a lower priority than the current one, based on the priority rank of the customer, the revenue it produces etc. and the orders which are not started at the moment or on hold at the time of scheduling.*

  The main purpose of this is to find the feasibility of completing the shop order early, and by rescheduling other shop orders according to the above criteria, it will ensure that all production orders will be carried out properly.

- In addition, the system will also provide a functionality to provide dynamic scheduling according to the user requirement on each work day: This is to fine-tune the quality of the schedule using parameters such as operations that cannot be carried out because of machine failures which will be scheduled on different machines or postponed depending on the priority and the operations order, operations that cannot be carried out due to the unavailability of the parts required etc.

- Finally, the user will be given feedback on the changes made to the production schedule by the system and the reasons in order to do so; which will allow them to either conform to the changes done or revert them if necessary.

4.5.1. Agent Types

Following listed are the main agent types available in the Dyno Scheduling system:

- **Work Centre Agent**
  - Responsible for handling different work centres by assigning operations related to shop orders to specific work centres, keeping track of the progress of operations, assigning relevant resources etc.

- **Shop Order Agent**
  - Accountable for shop orders and ensuring that the shop orders are delivered on the required date.
  - Keep track on the progress of the operations and the order in general by communicating with the work centre agent and negotiate with the resource agent when assigning resources to shop orders.
- **Manager Agent**
  - Assigned to managing all the above type of agents and facilitating their requirements.
  - Initiate the entire process at the beginning when there are shop orders, resources, work centres etc. added to the system or removed from the system.

4.6. Features

Listed below are the main features of the DynoSchedule system:

- Initial scheduling of production orders for a specified period of time-interval.
- Using real-time information for dynamic rescheduling of orders, maintaining the need dates and optimal utilization of available resources.

- Using a Prioritized-Adaptive Scheduling methodology to reduce the amount of orders that will be rescheduled which will be easier for the user when interacting with the system.
- Providing a user-friendly interface to interact with the system and provide proper feedback to the user regarding the decisions taken by the system.

4.7. Users

The proposed DynoSchedule system can be utilized by different types of users such as:

- *Production Planners*
    - Production planners develop the production schedule details the relevant operations and the work centres, monitor the progress of the operations and perform changes to the operations when a disruptive event occurs.
- *Shop-floor Supervisors:*
    - Supervisors perform front-end related operations such as opening, closing operations, delegating tasks to employees, monitoring work centres etc.
- *Shop-floor Labours*
    - Shop floor labours are assigned with daily operations that should be carried out on a particular work centre depending on their skill-level.

4.8. Summary

In this section, the most important part of this thesis was discussed consisting of the hypothesis, basic inputs, main outputs, identified users such as the planners, supervisors and labours, main processes and features of the proposed DynoSchedule system etc. In the upcoming chapters, these details will be elaborated and discussed in-depth. Accordingly, in the next chapter, design details of the DynoSchedule system are discussed including the high-level architecture of the system, class and database diagrams as well as the novel concepts introduced in the DynoSchedule system.

Chapter 5

Design of the Multi-Agent Based DynoSchedule System

5.1. Introduction

In this chapter, the design of the proposed DynoSchedule system will be discussed in detail. Initially, the higher-level architecture of the system will be discussed and afterwards, the main design details considered for the evaluation of shop orders when scheduling will be elaborated. Later, the Class Diagram and the Database Diagram of the system will be illustrated and finally, the novel concepts introduced in the DynoSchedule system will be detailed with regards to conventional and existing Multi-Agent based approaches for scheduling.

5.2. Architecture of the System

Following displayed is a high-level architecture of the proposed system:

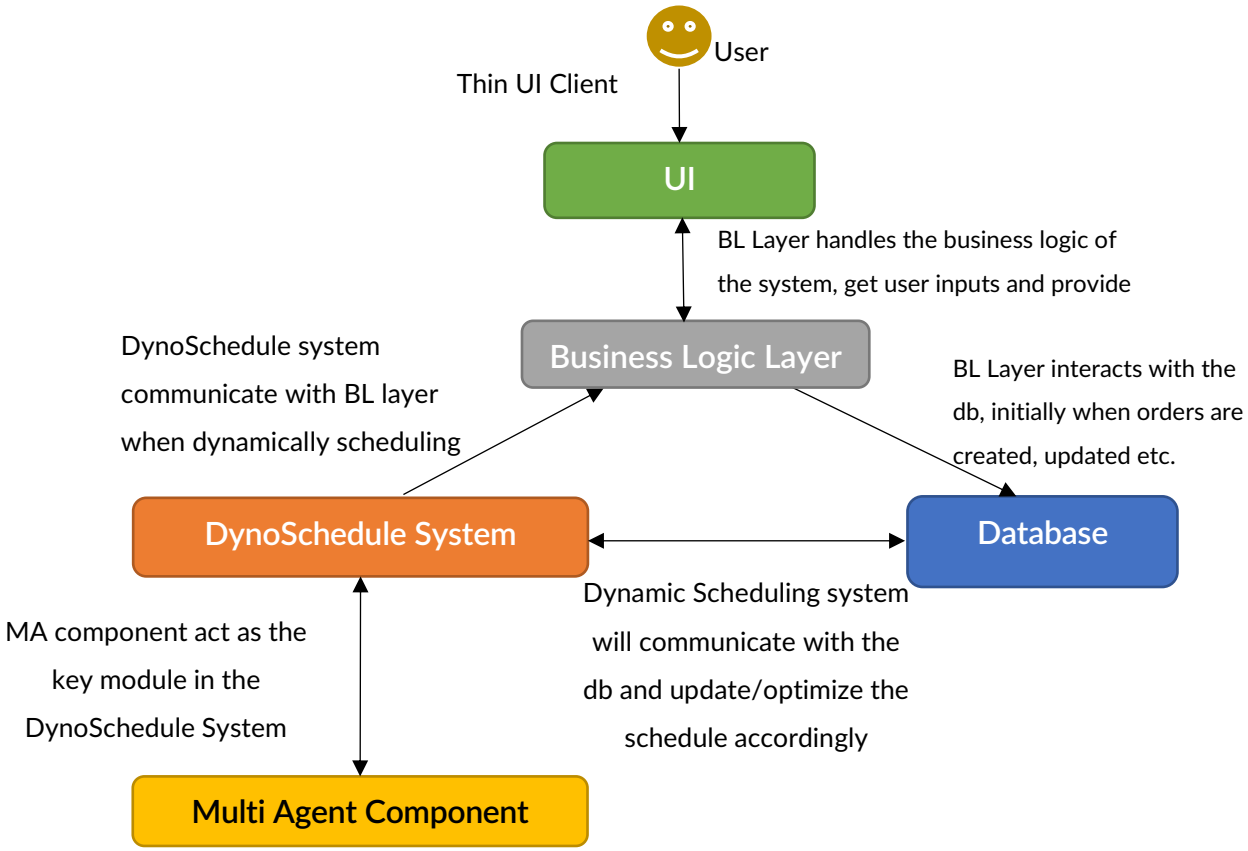


Figure 15.High-level architecture of the System

Figure 15 illustrates how the users can interact with the system through the User Interface and perform various functions such as inserting new production orders to the system, view the current schedule of the orders in a timeline view, dynamic rescheduling on as-needed basis, view different feedback given from the system etc. The UI is developed thin and loosely coupled from the Business Logic layer.

The Business Logic layer, as the name implies, acts as a middle layer and handles the business functions of the system and communicates with the DynoSchedule system and the database when necessary.

The database is currently managed by a relational database management system using MySQL, where the relevant models are maintained.

Finally, the DynoSchedule system works as the backbone of the main system, which comprises of a Multi Agent component. It contains agents who access a common ontology of knowledge, rules and practices of constraint-based scheduling. The agents consist of types such as:

- Work Centre Agent
- Shop Order Agent
- Manager Agent

5.2.1. Work Centre Agent

This type of agent is responsible for handling different work centres by assigning operations related to shop orders to specific work centres, keeping track of the progress of operations, assigning relevant resources etc.

Properties

- Work Centre No.
- Work Centre Type
- Work Centre Description
- Work Centre Capacity

Functions

- Reserve work centre time for operations.
- Identify work centre no/s based on the work centre type.

- Allocation of work centre for operations:
    - If Forward Scheduling; start from the earliest possible date and move forward, provide the current date as the beginning and search forward.
    - If Backward Scheduling; start from the need date and schedule backwards, provide the required date as the end of the last operation and search backwards.
- Work Centre Agents will be created for each Work Centre available depending on the work centre types that should be used for the scheduling of operations.

5.2.2. Shop Order Agent

Shop Order Agents are responsible for managing the shop orders and ensuring that shop orders are delivered on or before the required date. The shop order agent will perform scheduling on operations and the order in general by communicating and negotiating with the Work Centre Agent as well as the Manager Agent.

Properties

- Shop Order Details, including its operations and material requirements
- Shop Order importance
- Scheduling direction

Functions

- Coordinating with work centre agents and scheduling shop order operations.
- Keeping track on the progress of the operations.

5.2.3. Manager Agent

Manager agent is assigned with the task of managing all the above type of agents and facilitating their requirements. Manager agent will initiate the entire process at the beginning when there are shop orders to be scheduled due to new additions, work centre interruptions, part unavailability etc.

Functions

- Assigned with the task of managing all the above type of agents and facilitating their requirements.

- Initiate the scheduling process at the beginning when there are shop orders, resources, work centres etc. added to the system or removed from the system.
- Identify the orders where there are operations that need to be scheduled due to reasons such as:
    - Part Unavailability
    - Work Centre Interruption etc.,

  and initialize relevant Shop Order and Work Centre Agents.

- Be a coordinator for the negotiations that occur between Shop Order Agents and the Work Centre Agents.

Figure 16 displays is a diagram that illustrates how the different agents mentioned above will interact in the system to perform dynamic scheduling.

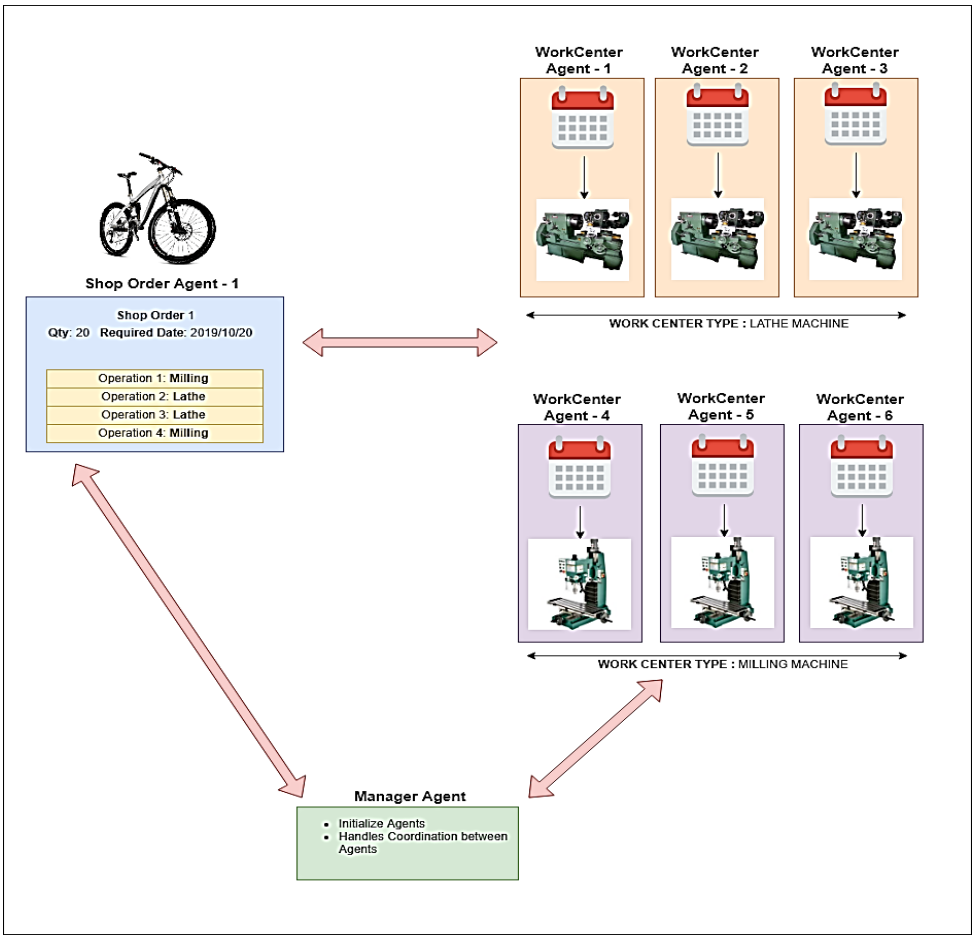


Figure 16.Agent Interaction

5.3. Shop Order Evaluation Criteria

Following discussed are the main criteria that is used to evaluate shop orders and operations when scheduling. These are divided into 2 main categories depending on the time that they are evaluated; One Time Criteria and Recurrent Criteria.

5.3.1. One Time Criteria (OTC)

One Time Criteria are used only once when initially scheduling the shop orders and the respective operations.

- *Shop Order Priority*
    - Priority given to each and every shop order based on the customer. There will be 5 prioritization levels depending on how long the customer has worked with the organization, the size of the customer, and the relationship between the customer and the organization

- *Shop Order Quantity: Income*
    - The quantity of the order, or the income that is generated by the order is a key factor when initially undertaking and scheduling the order and calculating the required date, or when deciding whether the shop order can be delivered by the customer required date.

- *Forward and Backward Scheduling*
    - When initially scheduling the operations, it will identify the direction from which the operations should be scheduled and do it accordingly.

5.3.2. Recurrent Criteria (RC)

Recurrent Criteria (RC) are used when dynamically scheduling the existing operations on an as-needed basis:

- *Availability of the Inventory Parts*
    - Similar to employee attendance, if the inventory parts that are needed for the operations are missing, it cannot be carried out. In such scenarios,
        - A different operation can be carried out early on that work centre if possible.

- *Availability of the Work Centres*
    - If the work centre is not available on a given time period (due to breakdowns or maintenance work), the operations that were scheduled on that time period to be carried out on that work centre;
        - Depending on the priority of the order, it will be assigned to a work centre, which is either vacant or has a lower prioritized operation scheduled on it.

- *Parallel and Sequential Operations*
    - Some operations can be carried out parallel, while others need prior operations to be finished. Scheduling will also consider this factor when doing the rescheduling.

- *Latest Operation Start Date (LOSD):*
    - Each operation is assigned with a LOSD by the system which is a critical factor that is used when scheduling the operations dynamically. In all cases, an operation should be scheduled on or before the LOSD calculated by the system.

- *Required Date*
    - Required date of the order is considered when dynamically rescheduling along with its prioritization level.

5.4. Class Diagram

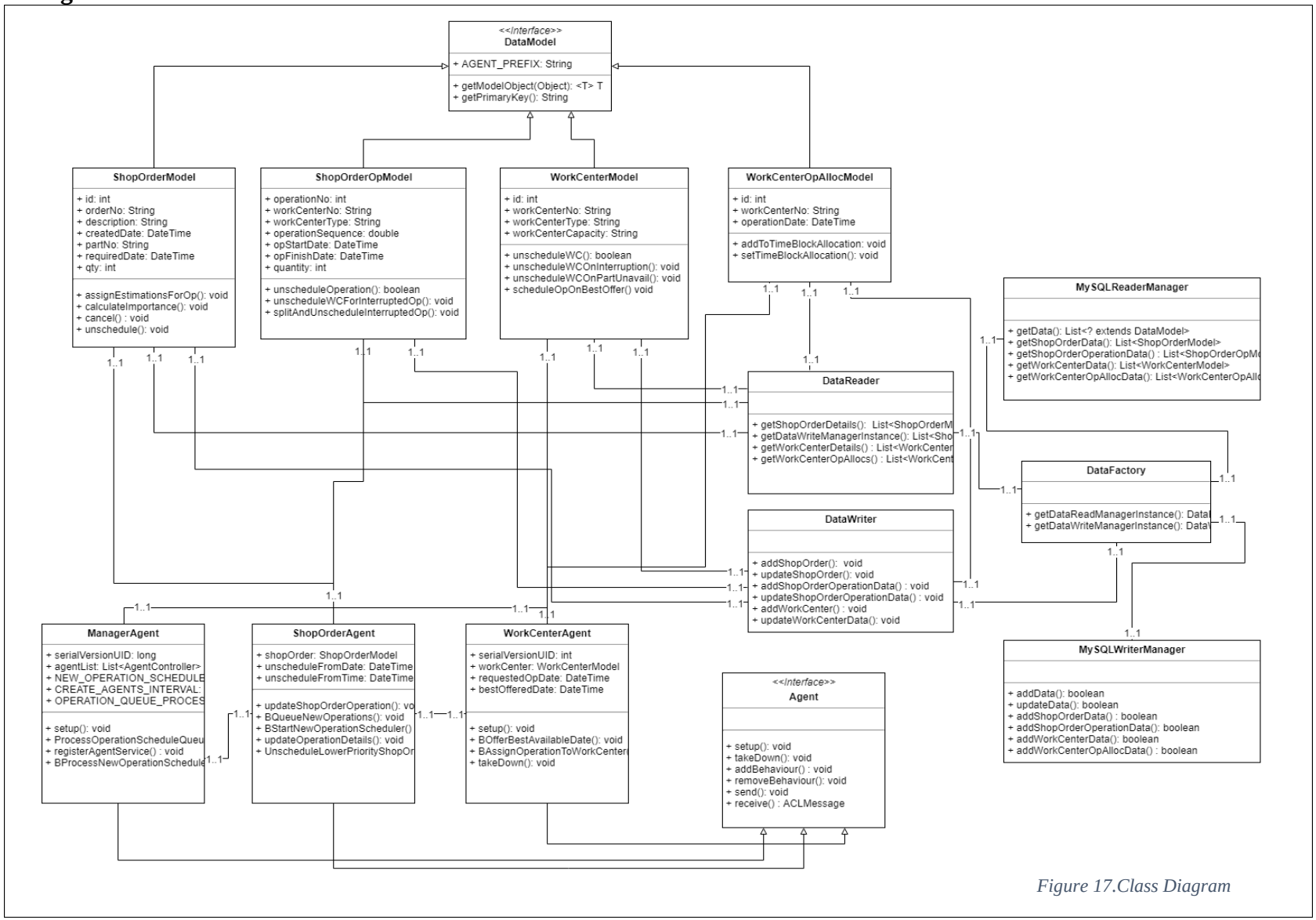


Figure 17.Class Diagram

5.5. Database Diagram

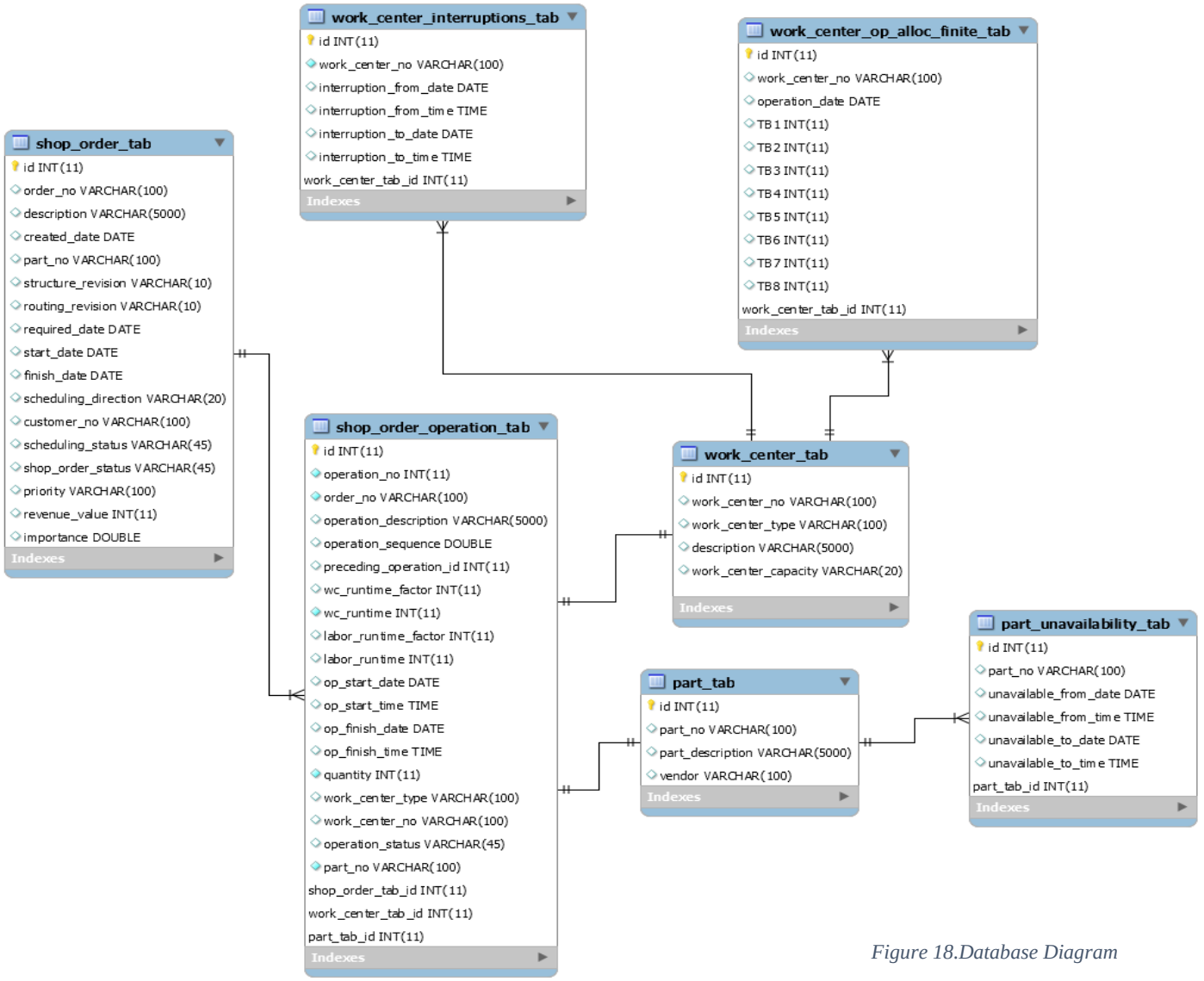


Figure 18.Database Diagram

5.6. Novel Concepts Introduced in the DynoSchedule System

There are 2 main novel features of the DynoSchedule system when compared to both conventional scheduling systems as well as available Multi-Agent based scheduling systems:

- **Prioritized-Adaptive Scheduling Concept**: In current Adaptive Rescheduling Concepts, Rescheduling process occurs for the times affected on the schedule, and a chained rescheduling process will happen for as long as all the conflicted times are resolved in the schedule. This approach leads to a large amount of orders to be rescheduled, which in a complex organization, takes a great amount of time and leads to rapid outdate of the production plan. In DynoSchedule system, it introduces an extension to the Adaptive Scheduling conception, by minimizing the number of rescheduled orders in the scheduling chain. This is achieved mainly by 2 ways:

    - The system calculates the upper limit of a particular operation start date known as the ***Latest Operation Start Date (LSOD)*** and the Shop Order Agent greedily tries to schedule a given operation on or before that date by negotiating with a Work Centre Agent. As long as the Shop Order Agent can schedule operations by meeting that point of time, it will keep scheduling the subsequent operations, however as soon as one operation fails to meet its LSOD, all the subsequent operations from that operation will also fail to meet their respective LSOD hence the SO Agent will stop scheduling from that operation onwards and try to unschedule lower priority shop orders and continue the scheduling process yet again.

    - Each shop order is given an "importance" score according to its customers priority and the revenues that are generated by them, using a weighted average method, which allows the manufacturing organization to prioritize them. These scores are then used for Prioritized Adaptive Scheduling of the generated schedule when a disruptive event occurs; these scores are evaluated by the Manager Agent and only the orders that have received a lower priority score than the current order and is closest to the current order will be used in the chain event, and using a cyclic approach, the system will try to come to an equilibrium as soon as possible with the least amount of orders affected when dynamically rescheduling.

This method is much efficient than the existing Adaptive Scheduling approaches since the amount of orders and the operations affected will be much less when the system comes to an equilibrium.

- **Advanced Market-like negotiation mechanism**: This negotiation mechanism is used by the main Agents associated with the DynoSchedule system, which provide a highly dynamic and scalable approach to scheduling. There, as mentioned previously, the Shop Order Agent uses the system calculated LOSD in order to keep the order schedule on track when engaging in the market-like negotiation mechanism with different Work Centre Agents.

5.7. Summary

In this chapter the design of the system was discussed by emphasising on the Architecture of it in terms of both a higher-level as well as a lower-level, indicating how the different agents are designed and their functionalities etc. In addition, the novel concepts introduced in the DynoSchedule system compared to existing dynamic scheduling systems as well as conventional scheduling systems are also discussed. Furthermore, the design of the system has also been illustrated using different diagrams such as the Class Diagram and the Database Diagram.

Chapter 6

Implementation of the DynoSchedule System

6.1. Introduction

In this chapter, the implementation particulars of the system are discussed elaborately. Initially, the different types of agents available in the system are discussed such as the Shop Order Agent, Work Centre Agent and the Manager Agent. Afterwards, the implementation details of the system will be discussed exhaustively using flow charts, sequence diagrams etc. for different functions of the system such as the initial scheduling process and the Prioritized-Adaptive Scheduling process when parts are unavailable, or work centres are interrupted.

6.2. Shop Order Agent

Following is a flowchart for the Shop Order Agent's initiation process.

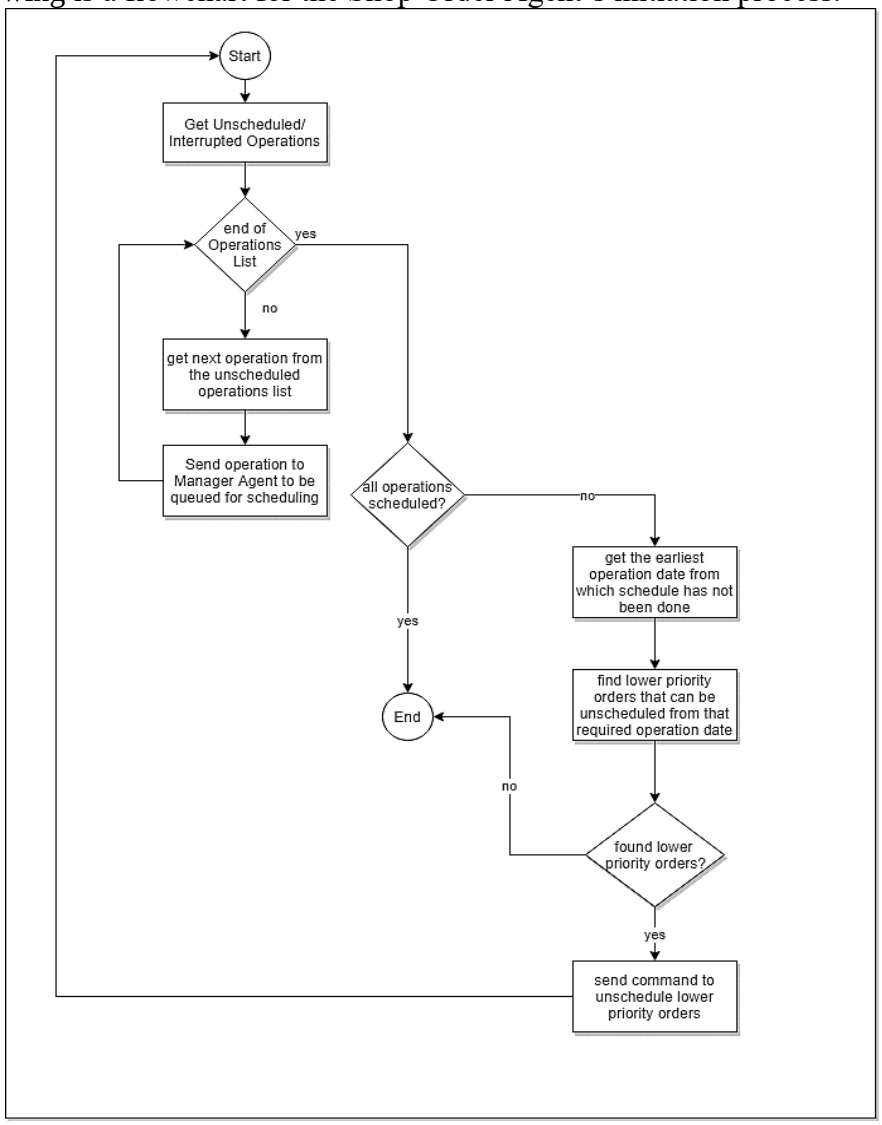


Figure 19. Shop Order Agent Initiation process

There, as visible in the figure 19, the Shop Order Agent will initially identify the operations (Unscheduled or Interrupted) that needs to be scheduled from its shop order and send requests to the Manager Agent for each of them. Per se, an active Shop Order Agent will have a list of operations that needs to be scheduled at any given time and will sequentially negotiate with the Work Centre Agents and try to greedily schedule them. There are 2 main scenarios associated with this schedule:

- ***All the operations can meet the Latest Operation Start Date:***

  For each of the operations, the system will calculate a **Latest Operation Start Date** which is the date and time before which the operation *must* start in order to meet the Shop Order's required date, because all the subsequent operations completing on the required time will depend on that. If this date or an earlier date can be negotiated with a Work Centre Agent, there will be no issues when scheduling the operation.

- ***An operation fails to meet the Latest Operation Start Date:***

  As mentioned previously, if one operation is unable to meet its Latest Operation Start Date (LOSD), this indicates that all the subsequent operations will not be able to meet their respective LOSDs as well, since they will depend on the current operation. In such a scenario, the Shop Order Agent will identify the **Earliest Operation Start Date (EOSD)** of such an operation (the first operation which fails to meet its LOSD) and stop scheduling all the subsequent operations. The assumption here is that, in order to meet the LOSDs with the current condition, the system will have to *move on to Prioritized Adaptive Scheduling mode*. As such, the Shop Order Agent will try to identify the Lower Priority Shop Orders (a maximum of 3 Shop Orders per iteration) that can be unscheduled from the identified EOSD, and then send a command to un-schedule them. Afterwards, the scheduling process will start again for the current shop order as well as the unscheduled shop orders for which, new Shop Order Agents and respective Work Centre Agents will be created by the Manager Agent, while being given the priority to the current shop order, so that it will be able to schedule its operations earlier than the current lower priority shop orders. In case if the shop order is unable to find any lower priority shop orders, that indicates that it is at the lowest priority of the system and all other shop orders have a higher priority. Then the planner will have to intervene and take an

administrative decision to either extend the required date or take other appropriate measures.

When the operation scheduling queue is processed by the Manager Agent, the Shop Order Agent will get an opportunity to schedule each of its operations by directly communicating and negotiating with the Work Centre Agents.

Figure 20 displays how this process is handled by the Shop Order Agent through a flowchart.

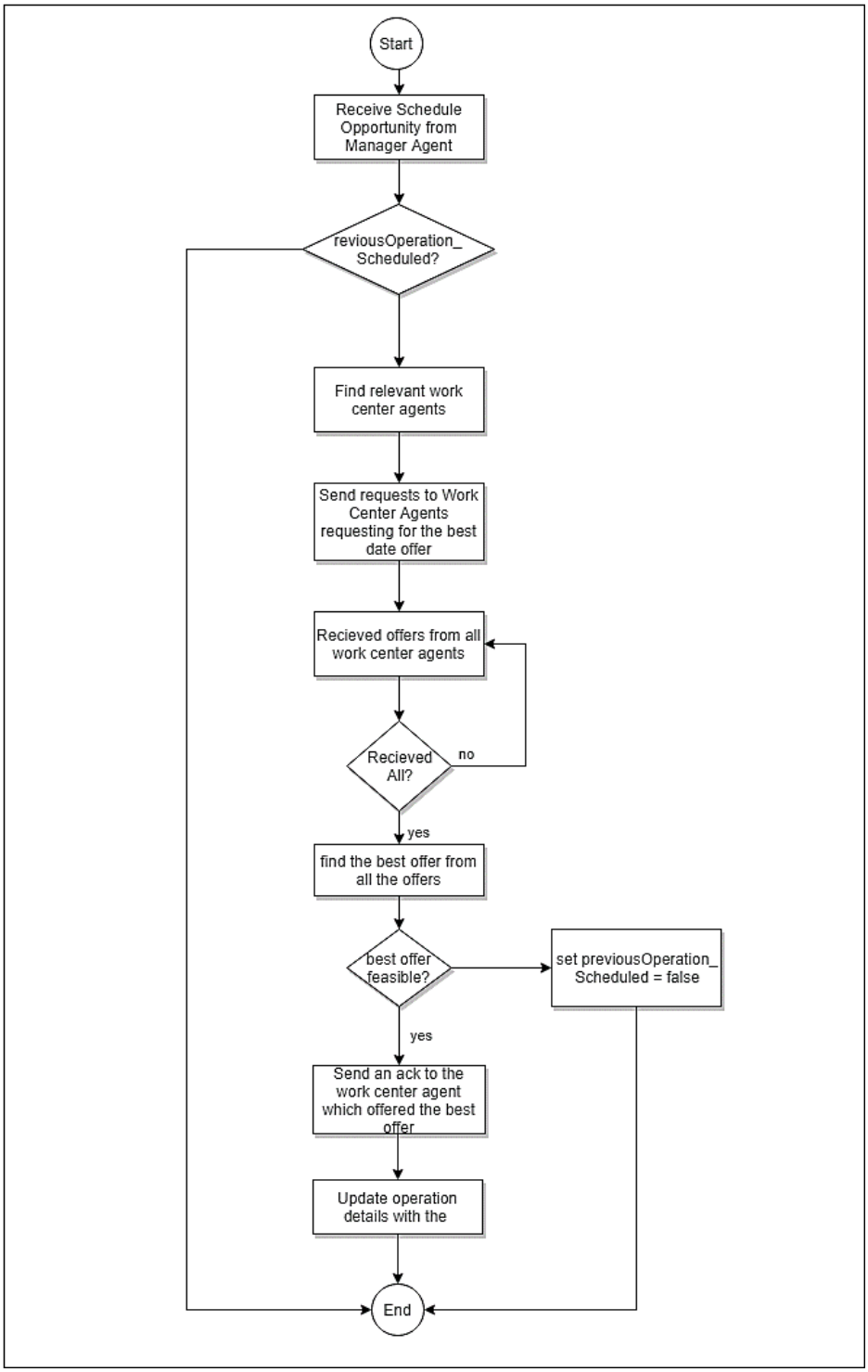


Figure 20. Shop Order Agent Operation scheduling process

Here, the Shop Order Agent will try to schedule each operation as it is provided the opportunity by the Manager Agent. Initially, the SO Agent will check if the previous operation (from the list of operations that should be scheduled on that instance) has been scheduled properly, which means that it has been able to meet its Latest Operation Start Date (LOSD) and if so, continue with the scheduling process for the current operation. If the previous operation was not able to meet its LOSD, that means, the current Operation should not be scheduled since it will not be able to meet its LOSD as well. Therefore, in such a scenario, the scheduling process will not happen for the current operation. If the previous operation has met its LOSD, then the current operation will go through its scheduling process. There, it will first send a message to all the respective Work Centre Agents by providing the EOSD and request for a feasible date, and after all the offers are received, it will select the best offered date and send an acknowledgement to the respective Work Centre Agent to schedule the operation. Subsequently, it will update the operation details with the Start Date, Finish Date and set its status to *Scheduled*.

This process is explained on a larger perspective, on how different Agents will interact, using a sequence diagram later.

6.3. Work Centre Agent

Displayed on Figure 21 is a flowchart for the Work Centre Agent providing the best offer to a Shop Order Agent and updating the relevant details in the database:

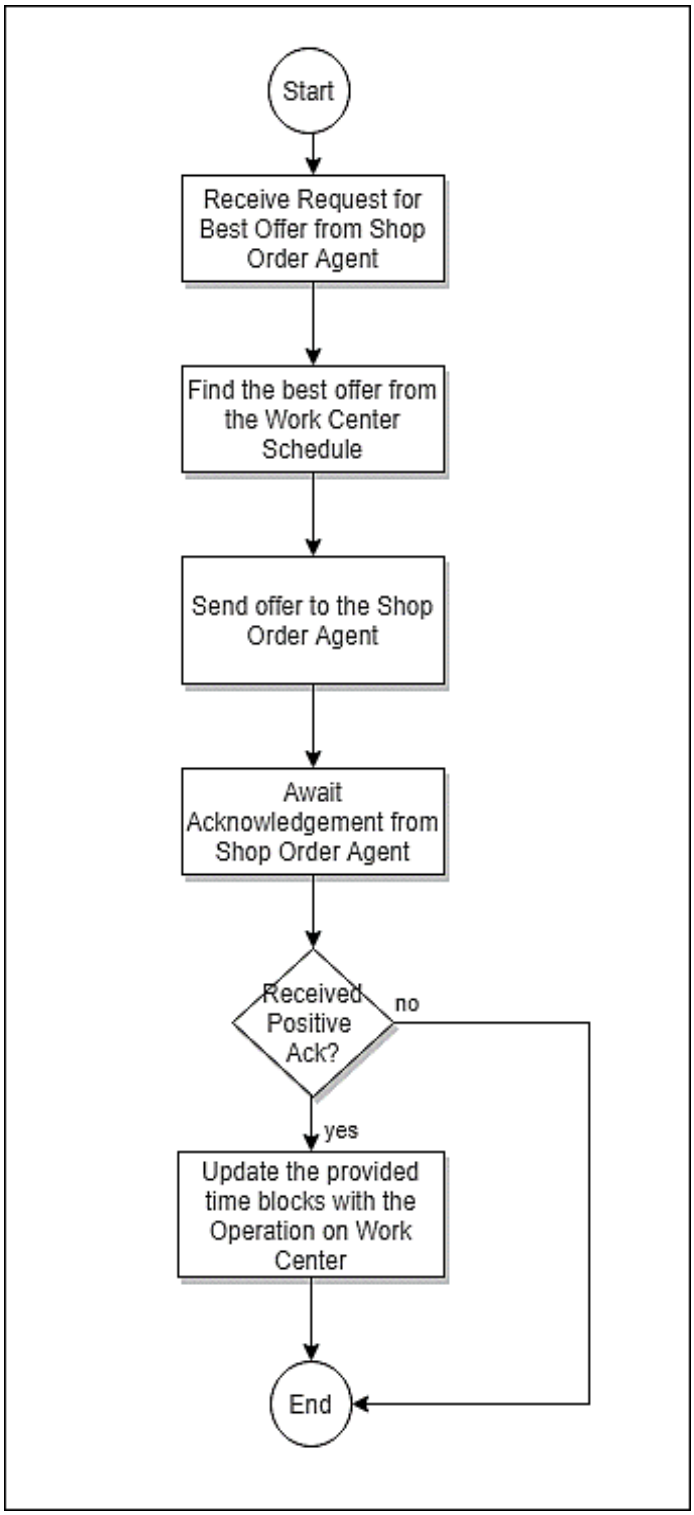


Figure 21.Work Centre Agent Providing the Best Offer

This is a relatively simple process in terms of a Work Centre Agent. After it receives an EOSD of an operation to be scheduled from a Shop Order Agent, the Work Centre Agent will execute a capability check within the business logic layer of the system to identify the best possible date that it can offer for that operation to be scheduled. There, several conditions will be taken into consideration when providing the best offer:

- Find the most feasible date that comes *on or after* the Earliest Operation Start Date provided by the Shop Order Agent.
- The operation should be scheduled without any interruption for its entire runtime, which is a requirement of the Manufacturing organizations in order to minimize the wastage and reduce the setup times of Work Centres (As a work-around for this, **Split** operation process can be used, which will be discussed in more detail later in the study).
    - When considering this criterion, the work centre will first check if there are any planned interruptions such as maintenance work or downtimes for the work centre.
    - In addition, it will also check if the part that will be used by the operation is available during that time period where the operation will run on the work centre.

After the best date that can be offered is identified by the Work Centre Agent, it will communicate that to the Shop Order Agent, and wait for its proposal acceptance. If and when the Shop Order Agent accepts the Work Centre Agent's proposal, it will receive a message and upon receiving, the WC Agent will update its work centre schedule and reserve the date from the offered date and time for the operation, throughout its runtime.

6.4. Manager Agent

Manager Agent acts as the main initiator as well as the coordinator of all the agents. It is assigned with the task of initializing agents depending on the shop orders that need to be scheduled and the related work centres, coordinating the requests to schedule operations from shop order agents, taking down agents after the work is completed etc.

Displayed below are flow charts illustrating the functionality of the Manger Agent:

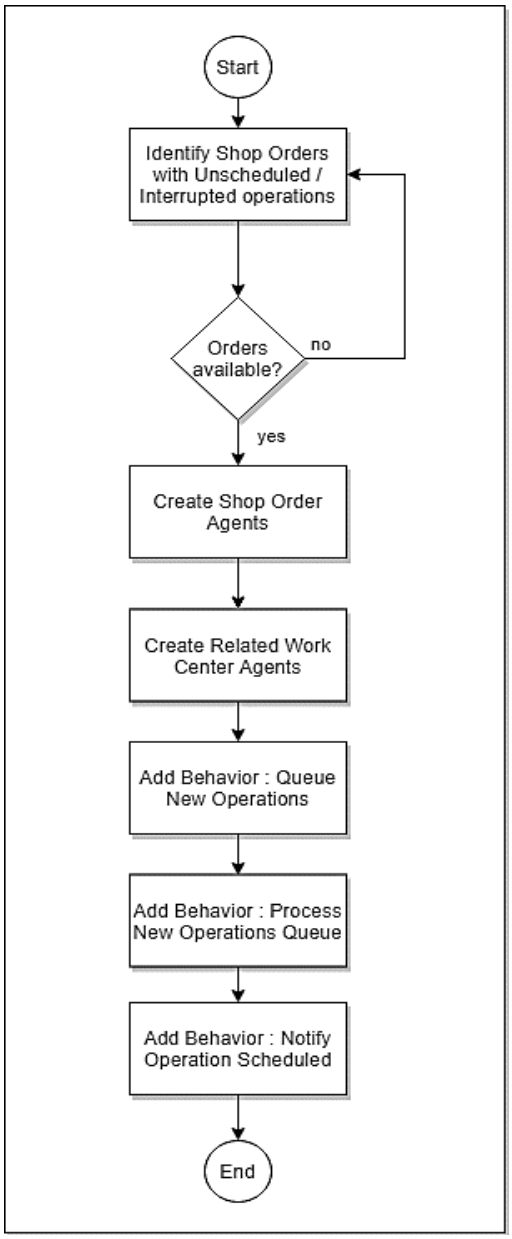


Figure 22.Manager Agent Initiation

Figure 22 displays how the Manager Agent initiates the different Agents and add behaviours to itself accordingly.

Manager Agent is the only type of Agent who will always be active on the system, thus it will run on a Timer behaviour and create Shop Order Agents and the respective Work Centre Agents to schedule the orders and their operations. There, the Manager Agent will keep on identifying operations with *Unscheduled* or *Interrupted* status and their respective Shop Orders on a regular interval. Subsequently, it will create the Agents for those Shop Orders as well as the Work Centres, and then, start its behaviours to accept Operation Schedule Requests, process them and to Lock/Unlock the operation schedule queue.

Figure 23 illustrates the short process of queueing the shop order operation requests received from a Shop Order Agent.

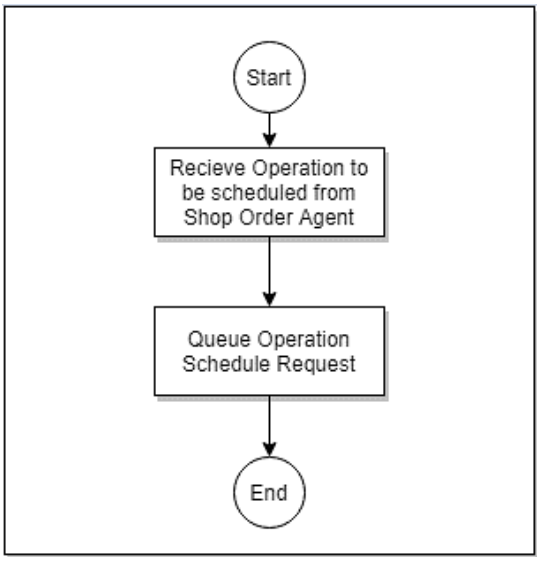


Figure 23.Queuing new operations to be scheduled

Figure 24 illustrates how the queue is processed in order for the operations to be scheduled. After the shop order operation requests are received from a Shop Order Agent, as mentioned, it will put them in a queue to be processed. Afterwards, it will sort the queue based on the Priority of the Shop Order, and the Sequence No. of the operations. Finally, for each of the operations in the queue, it will pop them and send a message to the Shop Order Agent indicating that it can start schedule on that particular operation, and will lock the thread to schedule other operations until an acknowledgement is received from that particular Shop Order Agent. The main reason to have this locking mechanism on the Manager Agent is that the Shop Order Agents and Work Center Agents work on a market-like negotation mechanism, where the SO Agent can accept any of the offered dates by the WC Agents, and they should be able to accept and reserve that time period if the SO Agent accepts it. In order to do so, having a sequential

mechanism to process the operation schedule requests is vital or otherwise the WC Agents will face issues when accepting the best offered date on a parallel processing scenario where it provides the same or overlapping dates as the best offer to multiple Shop Order Agents and they all accept those offered dates. Therefore, the Manager Agent has been employed as a coordinator between the Shop Order Agents and the Work Center Agents to make the scheduling process execute more smoothly, in a sequential manner.

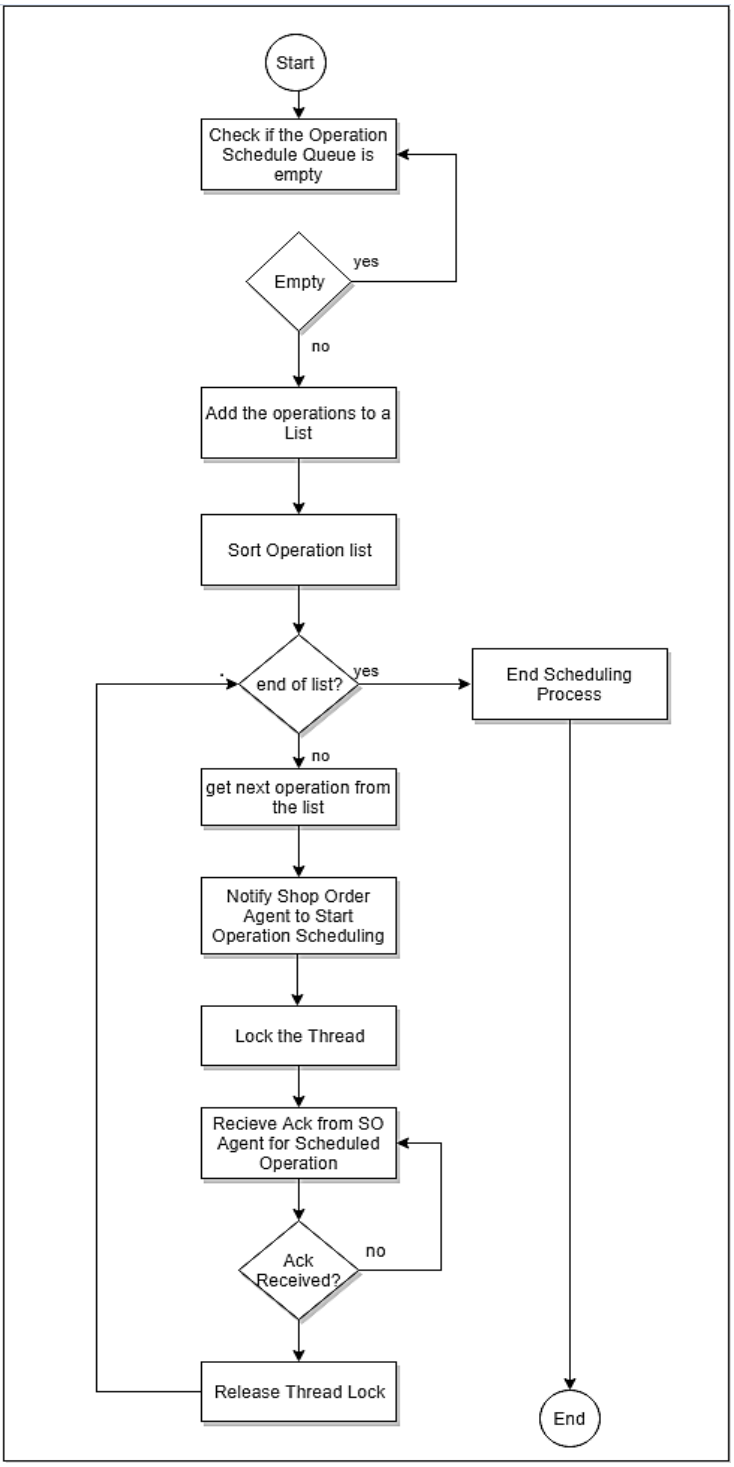


Figure 24.Processing Operation Schedule Queue

6.5. Implementation of the System

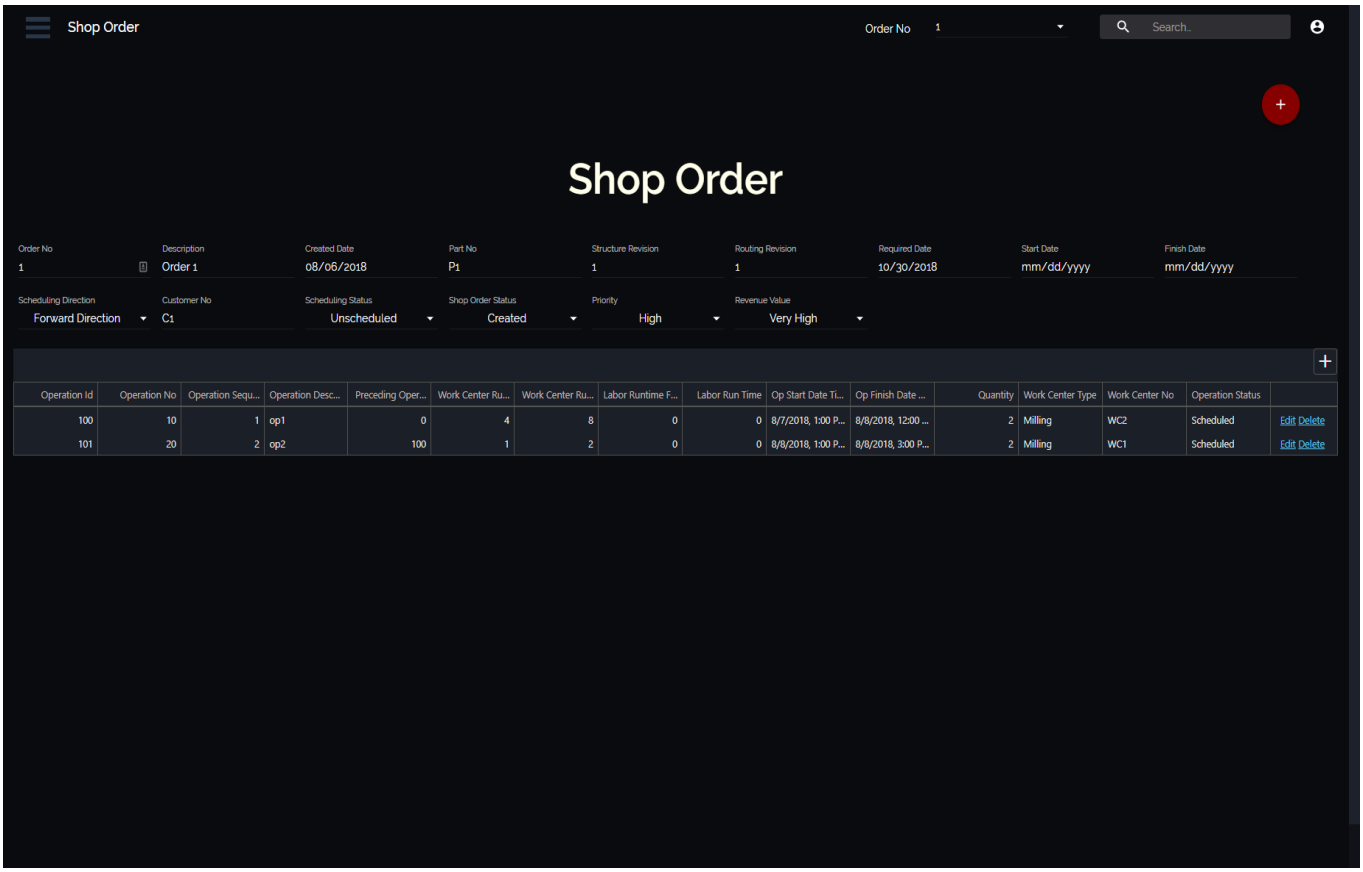


Figure 25. Shop Order Details UI

In order to enter a Shop Order to the system, the user can make use of the Shop Order Details UI. Figure 25 displays a screenshot of the Shop Order Details UI developed for the DynoSchedule system.

Following are the main features provided from the Shop Order Details UI:

- Create Shop Orders
- Update Shop Order Details
- Cancel Shop Order
- Create Operations for existing Shop Orders
- Update Operation details
- Change the Operation statuses of new Shop Orders to *Unscheduled:* When inserting a new operation, by default, it will be set to *Created* status, upon which the user can do any number of updates on the operation and the Manager Agent will not consider such operations to be scheduled. In order for the Manager Agent

to consider those operations, their statuses should be changed to *Unscheduled.* This can be done from the UI just by a click of a button.

Figure 26 shows a sequence diagrams of the agent interaction that happens during this process:

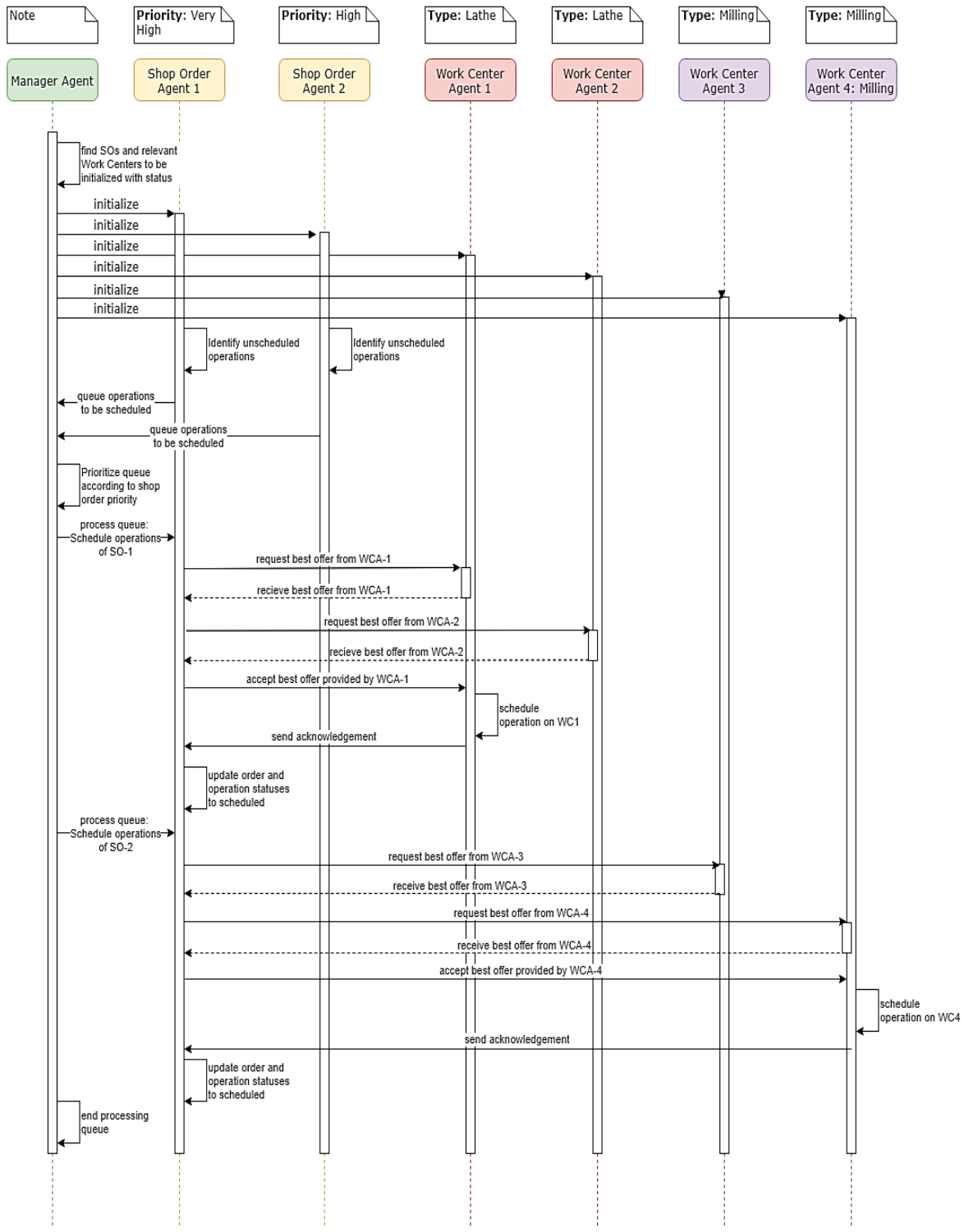


Figure 26.Sequence Diagram for Initial of Orders

Initially, the Manager agent will identify the orders that need to be scheduled and the related work centres that are needed for that process and initialize the relevant Shop Order Agents and Work Centre Agents. Afterwards, the Shop Order Agents will identify the operations that they need to schedule and communicate them to the Manager agent who will order them according to a priority queue. Afterwards, the Manager Agent will start processing the queue allowing operations to be scheduled one after another. The process includes an advanced market-like negotiation mechanism where the Shop Order Agents request all the relevant Work Centre Agents to provide their earliest start date for the operation to be scheduled by providing the Shop Order Agent's preferred start date for the operation. After receiving all the offers for the earliest start date from the Work Centre Agents, the Shop Order Agent will then compare the dates greedily and select the earliest possible start date, that doesn't pass the Latest Operation Start Date (LOSD) for that operation and send an acknowledgement to the relevant Work Centre Agent, who will then proceed to reserve the relevant period (which is the runtime of the operation) from its calendar to carry out that operation. This process will continue repeatedly until all the operations are scheduled. This is the most basic process of the system without using any adaptive scheduling aspects.

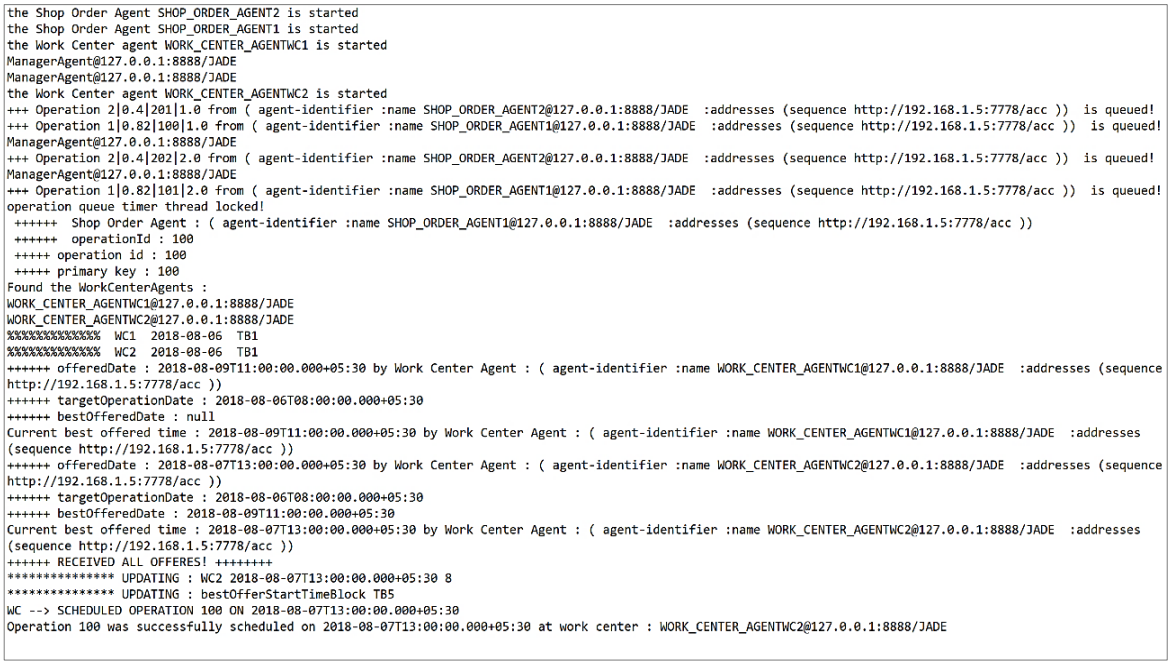


Figure 27.Sample Message Space of Agent Communication

Figure 27 shows a sample message space of the communication that happens between the Shop Order Agent and the Work Centre Agents when initially scheduling an unscheduled shop order.

6.5.1. Prioritized Adaptive Scheduling Algorithm

The above process will be extended if and when there's a scenario, where a particular Shop Order Agent is unable to schedule one or more of its operations starting from its Earliest Operation Start Date (EOSD) due to being unable to meet its LOSD. On such cases, the system will start its Prioritized Adaptive Scheduling algorithm. There, as indicated previously, The SO Agent, who is unable to schedule its operations, will try to find lower priority shop orders than itself (a maximum of 3 shop orders) which can be un-scheduled in order to provide its operations the chance to be scheduled from the EOSD. If such shop orders are found, the SO Agent will send a command to the Business Logic layer of the system to un-schedule such orders which will remove the reservations of the appropriate operations (checked from EOSD) from the respective work centre and render their statuses as Unscheduled or Interrupted. In case if EOSD happens in the middle of an on-going operation, the system will execute a process known as **Splitting** the operation, which will dynamically split such an affected operation into multiple operations. The main purpose of executing this splitting process, as mentioned previously, is that the runtime of an operation cannot be interrupted in the middle, thus by splitting the operation into smaller operations, this can be achieved on a dynamic un-scheduling scenario. After the lower priority shop order operations are un-scheduled, since the Manager Agent is constantly monitoring for such operations on a regular interval, the next time this check is carried out, it will discover these Unscheduled or Interrupted operations and their related shop orders, work centres and create Agents for them, including the higher priority Shop Order of which, the operations could not be scheduled.

After creating those Agents, the process to schedule the operations will start again and the Manager Agent will again act as the coordinator for operation schedule requests. There, since the higher priority operations will be sorted to the top of the queue, the lower priority operations will have to take their places on the queue after them.

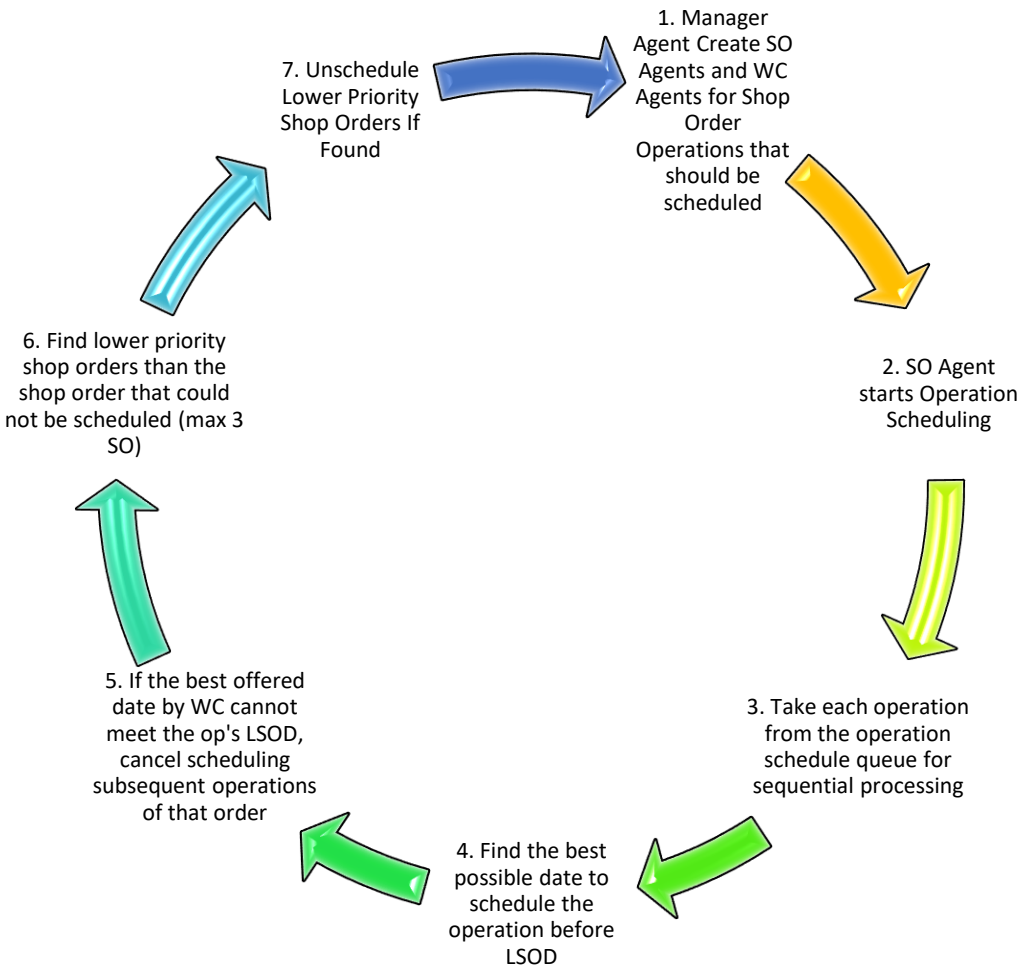


Figure 28.Prioritized Adaptive Scheduling Algorithm

Figure 28 shows the Prioritized Adaptive Scheduling algorithm in a high-level diagram.

There are certain benefits to the developed Prioritized Adaptive Scheduling algorithm when comparing with the existing Adaptive Scheduling approaches and following listed are some of them:

1. In the current Adaptive Scheduling algorithm, the un-scheduling chain is not restricted and will be carried out till all the affected operations are unscheduled, which will be a huge amount of list in a manufacturing organization. This will lead to chaos in the created manufacturing plans. However, in the Prioritized Adaptive Scheduling algorithm, the amount of orders that are un-scheduled is limited to a maximum amount of 3 per an iteration, and the shop orders will first try to achieve its Latest Operation Start Date (LSOD) rather than moving into

the un-scheduling process on each iteration, resulting in a much less amount of Shop Orders getting unscheduled.

2. With the no. of unscheduled operations being minimum, the amount of time taken to schedule those un-scheduled operations will be quite less when compared to the current Adaptive Scheduling algorithm and the Agents will come into equilibrium much quickly and efficiently.

3. Since the system will only affect the lower priority shop orders when un-scheduling, on a business perspective, the system helps the manufacturing organization to prioritize the orders that will provide more monetary benefits to it while de-prioritizing the orders with relatively less benefits.

4. In addition, since the system will not be able to continue its dynamic scheduling process if a shop order cannot find any lower priority shop orders than itself, rendering it as the lowest priority shop order, the organization management (including the Shop floor Planner or the Supervisor) will have the ability to take a managerial decision by discussing with the customer to change the required date of the shop order.

These kinds of abilities provided by the system will allow the organization to both get the maximum out of the system while reducing the chaos that might generate by the existing dynamic scheduling approaches.

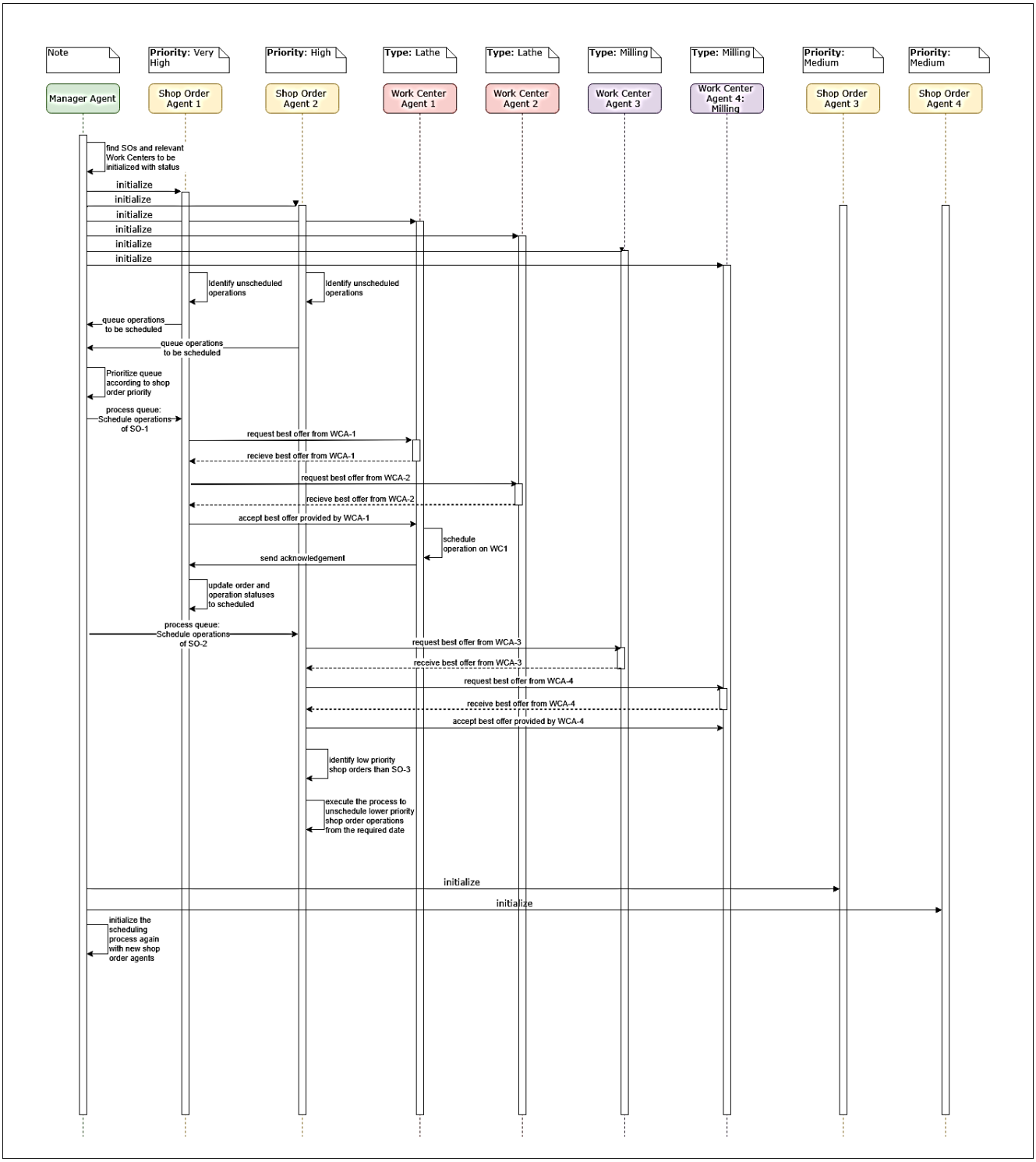


Figure 29.Sequence Diagram for Prioritized Adaptive Scheduling

Figure 29 illustrates a Sequence Diagram indicating the process of the Prioritize Adaptive Scheduling algorithm when the a given Shop Order cannot meet its LSOD initially and tries to achieve the scheduling by un-scheduling lower priority shop orders.

6.5.2. Prioritized Adaptive Scheduling based on Real Time Data

The DynoSchedule system will also provide the ability to carry out the Prioritized Adaptive scheduling process based on real-time data, per se, there are 2 important scenarios are considered.

6.5.2.1. Planned/Unplanned Work Centre Interruptions

It's quite common in manufacturing organizations to have interruptions on work centres which can be divided into 2 main categories:

- **Planned Interruptions:** Planned maintenance work, and downtime of work centres (machineries).
- **Unplanned Interruptions:** Breakdowns of machines.

When these kinds of interruptions occur on a work centre, the operations scheduled on that work centre will be affected and there should be a way to work around those interruptions. In such scenarios, again, the Prioritized Adaptive Scheduling process will intervene and allow for the affected operations to be scheduled again. There, after the user enters a Work Centre Interruption from the UI, the system will identify the operations that are affected by it. Afterwards, from the Business Logic Layer, it will un-schedule the operations, using the splitting process if necessary, within the interrupted time period of that work centre. Afterwards, the next time the Manager Agent executes the process to identify un-scheduled operations and related orders, it will catch these operations and create the related Shop Order Agents and Work Centre Agents to start the scheduling process. As such, the scheduling process will initiate and continue as explained previously. Accordingly, the core process of initiation, coordination and the negation between Agents will remain unchanged, yet the starting conditions of the Agents will be changed depending on the different inputs taken.

Figure 30 illustrates the Work Centre Details UI developed for the DynoSchedule system, which can be used to do various tasks such as:

- View Work Centre Details
- Add/ Update Work Centre Details
- Add Interruptions to the Work Centre
- View previous interruption details

As visible, a Work Centre can have multiple interruptions, and the user has the ability to add and view those interruption details from this UI as well.

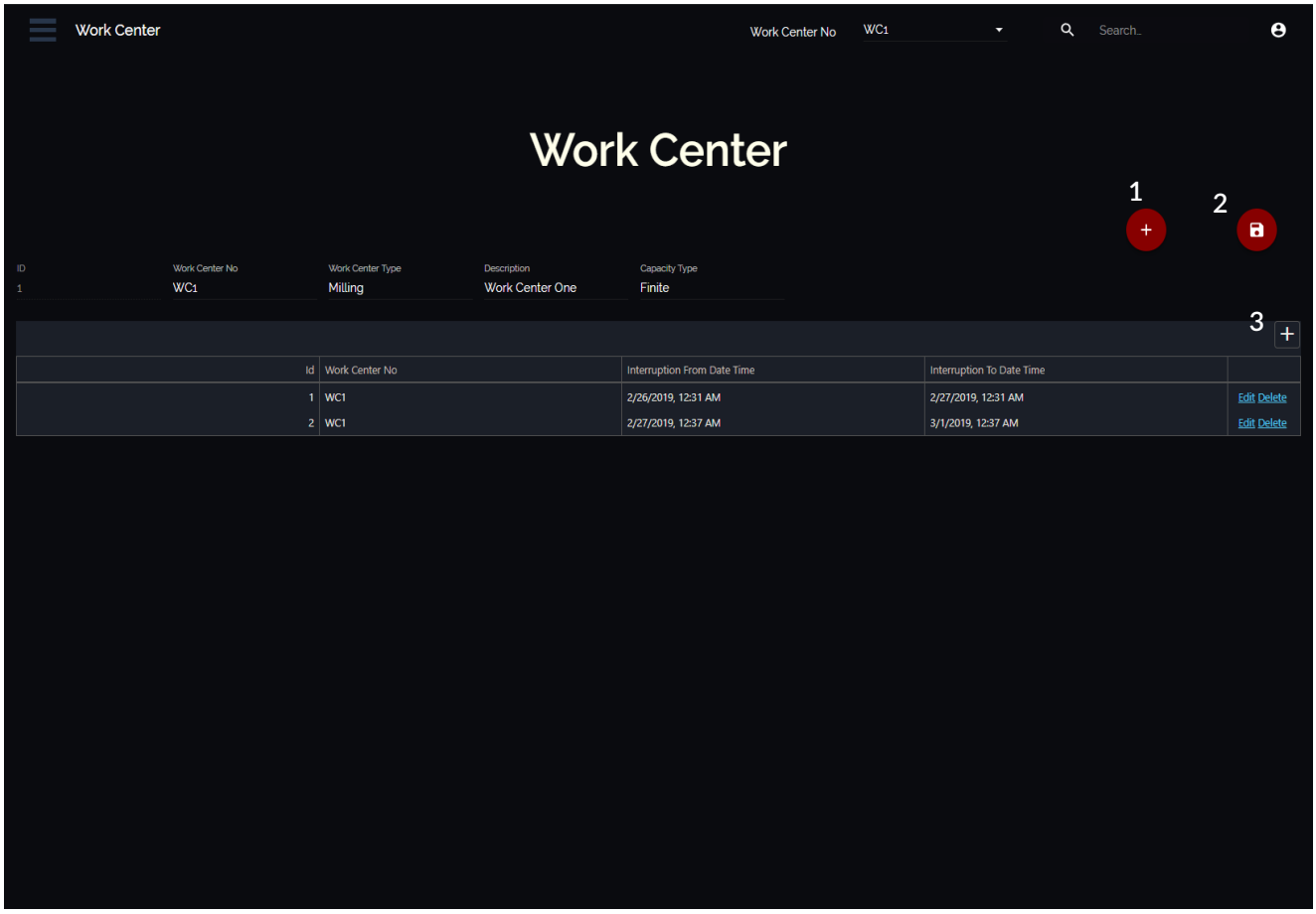


Figure 30.Work Centre UI

Following are the functions of buttons indicated on the Work Centre details UI as displayed on Figure 30:

1. ***Add Work Centre***: can be used to add new Work Centre details
2. ***Update Work Centre***: can be used to update the current Work Centre details
3. ***Add Interruption***: can be used to add new interruptions to the Work Centre

Figure 31 displays a sequence diagram for the Prioritized Adaptive Scheduling process with real time data for work centre interruptions

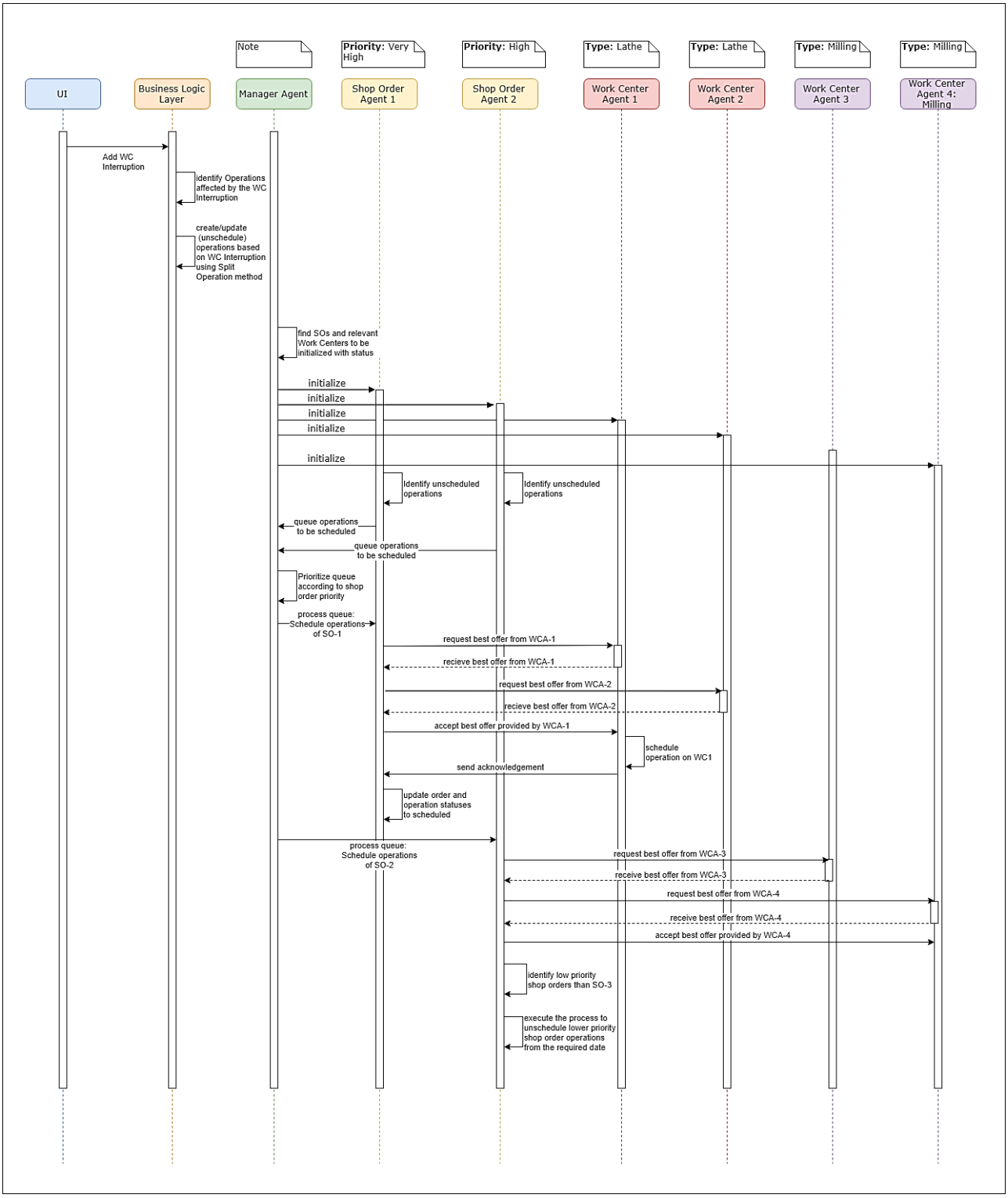


Figure 31.Sequence Diagram of Prioritized Adaptive Scheduling process with real time data for work centre interruptions

The details of the above process were discussed in greater detail at the beginning of the topic. Here, the core process of the Prioritized Adaptive Scheduling algorithm largely

remains unchanged, yet the starting conditions are changed using real-time information provided by the user.

6.5.2.2. Part Unavailability

Manufacturing organizations heavily depend on its suppliers to provide the parts that are required to carry out the manufacturing process. Per se, when the parts are not available to carry out an operation, this can hinder the progress of the entire shop order. There are various reasons for the part to be unavailable to continue an operation.

i. Part not being delivered on the required time by the suppliers.
ii. The delivered parts not being up-to the expected standard of the manufacturing organization rendering them unusable.
iii. The delivered parts being damaged or receiving a lower amount of the part than what is required.

Due to these kinds of reasons, the organization will have a lack of the relevant part to carry out the required operations for a specific period. In turn, this will affect the Work Centre schedule since, even if the operations are scheduled in the given period, they cannot be carried out due to the unavailability of the required part. Therefore, once these details are entered to the system on real time, the system should have the capability of adjusting the schedule accordingly.

However, when comparing with the Work Centre Interruption scenario, there is a difference in part unavailability scenario for dynamic scheduling. When a work centre is interrupted due to a particular reason, no operation can be carried out on that WC during that time. But for a part unavailability scenario, even though one operation cannot be carried out during that time due to the lack of parts, another operation can take its place because the Work Centre is still operational. This is a vital detail that should not be overlooked when dynamically scheduling the operations.

In this situation, after the user enters a part unavailability information to the system, again, from the Business Logic layer, it will identify all the operations that are affected due to the unavailability of the part, and it will un-schedule them from unavailability start date. There can be multiple orders, operations and work centres that are affected by this, which is in contrast with the work centre interruption which can only affect one particular

work centre. Therefore, DynoSchedule system handles this in a way that, after the un-scheduling process due to a part unavailability, the Work Centre will still have the ability to take different operations to the schedule during that time period.

When the Prioritized adaptive scheduling process starts again for the un-scheduled operations, and when the Shop Order Agents try to request the best offered date, the Work Centre Agents will, as mentioned previously, check for the availability of the work centre as well as the availability of the part during the time period. This will allow the Work Centre to not schedule the affected operations from the part unavailability during that period again, and schedule other operations on that time period as per the requirement.

Figure 32 displays the UI developed for the DynoSchedule system that will allow the users to do various tasks such as:

1. View part details
2. Update Part Details
3. View/Add Part Unavailability Details.

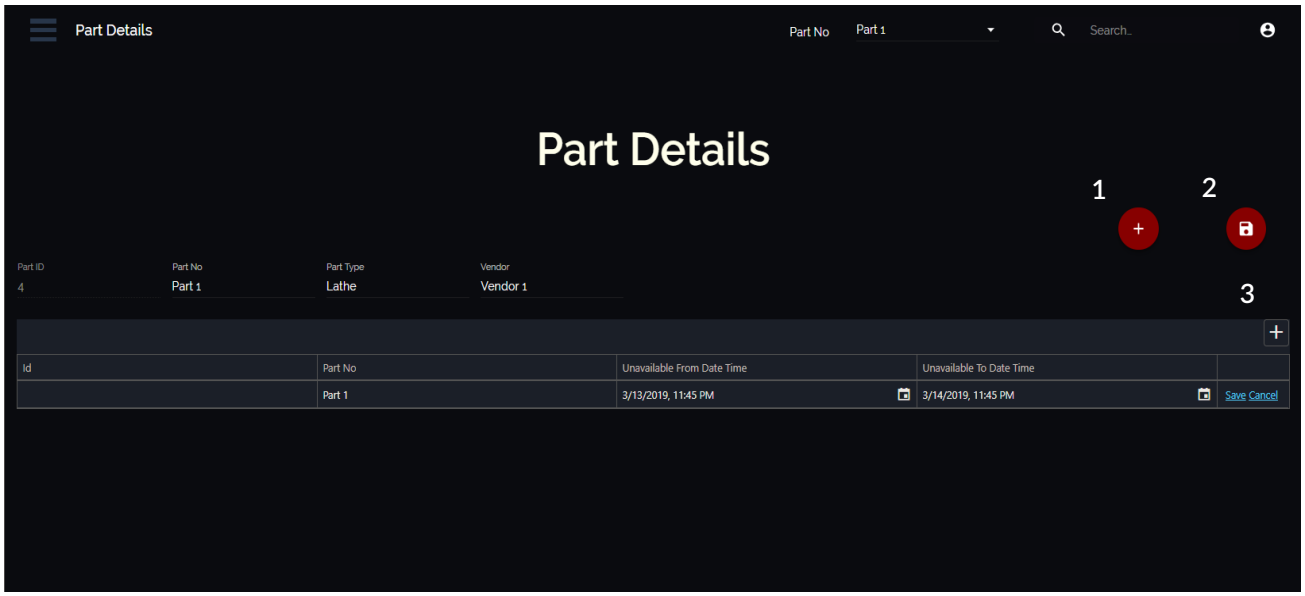


Figure 32. Part Details UI

Similar to the Work Centre Interruption details, Part Unavailability details also has a one-to-many relationship with the part details, meaning that one part can have many numbers of unavailability details on different dates and times.

Following are functionalities of the buttons indicated Part Details UI on figure 32:

1. ***Add Part details***: can be used to add new Part details
2. ***Update Part details***: can be used to update the current Part details
3. ***Add Part Unavailability***: can be used to add new unavailability detail to the Part

Figure 33 displays a sequence diagram for the Prioritized Adaptive Scheduling process for Part Unavailability scenario:

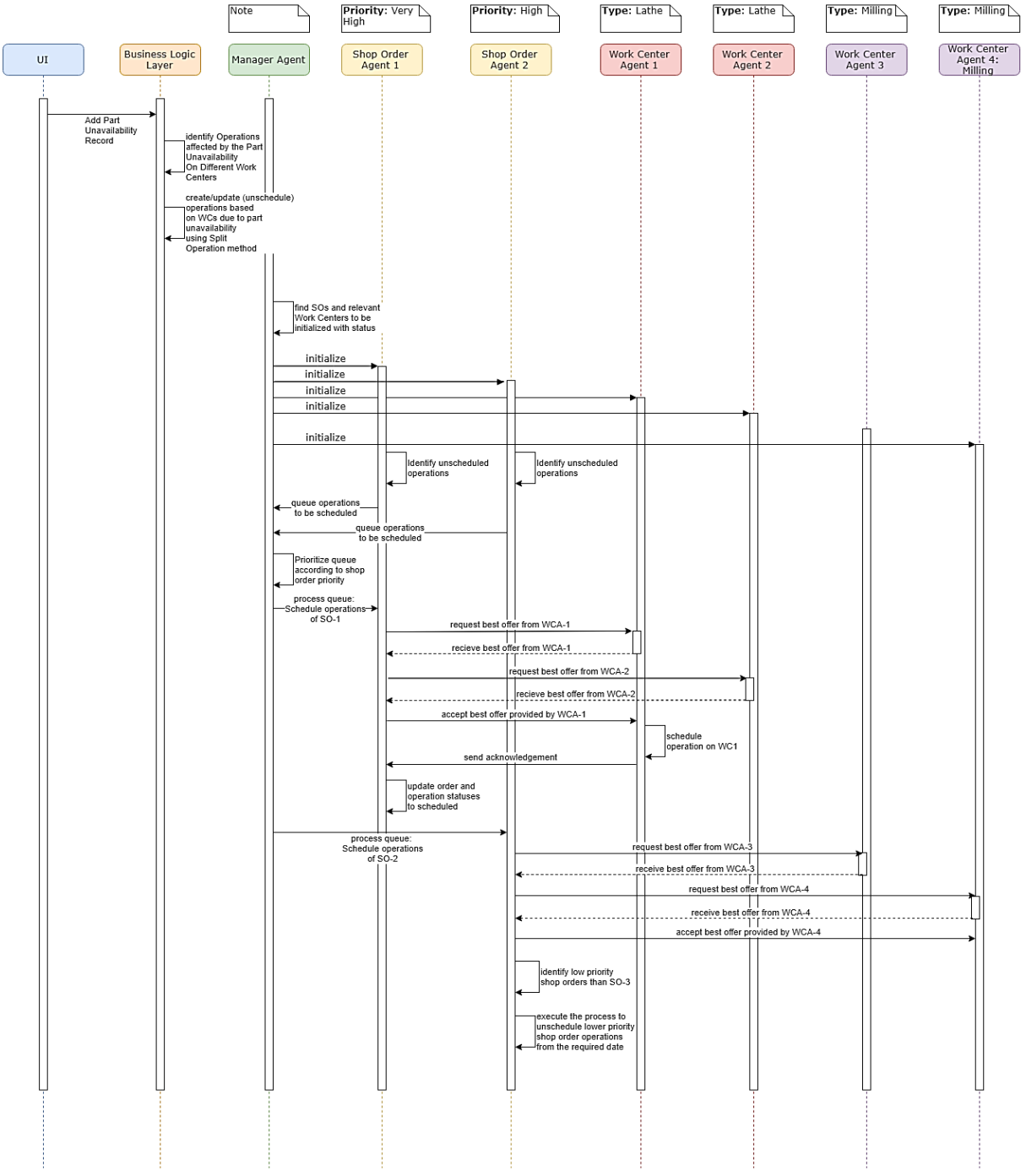


Figure 33. Prioritized Adaptive Scheduling process for Part Unavailability

6.5.3. Shop Order Schedule UI

After the Shop Order Operations are scheduled, the User will be able to use the Shop Order Schedule UI in order to view these scheduled operations. There, the operations will be grouped by the Order No, and additionally, they will also be grouped by the Work Centre on which they are scheduled on, using different colours that are generated dynamically from the UI.

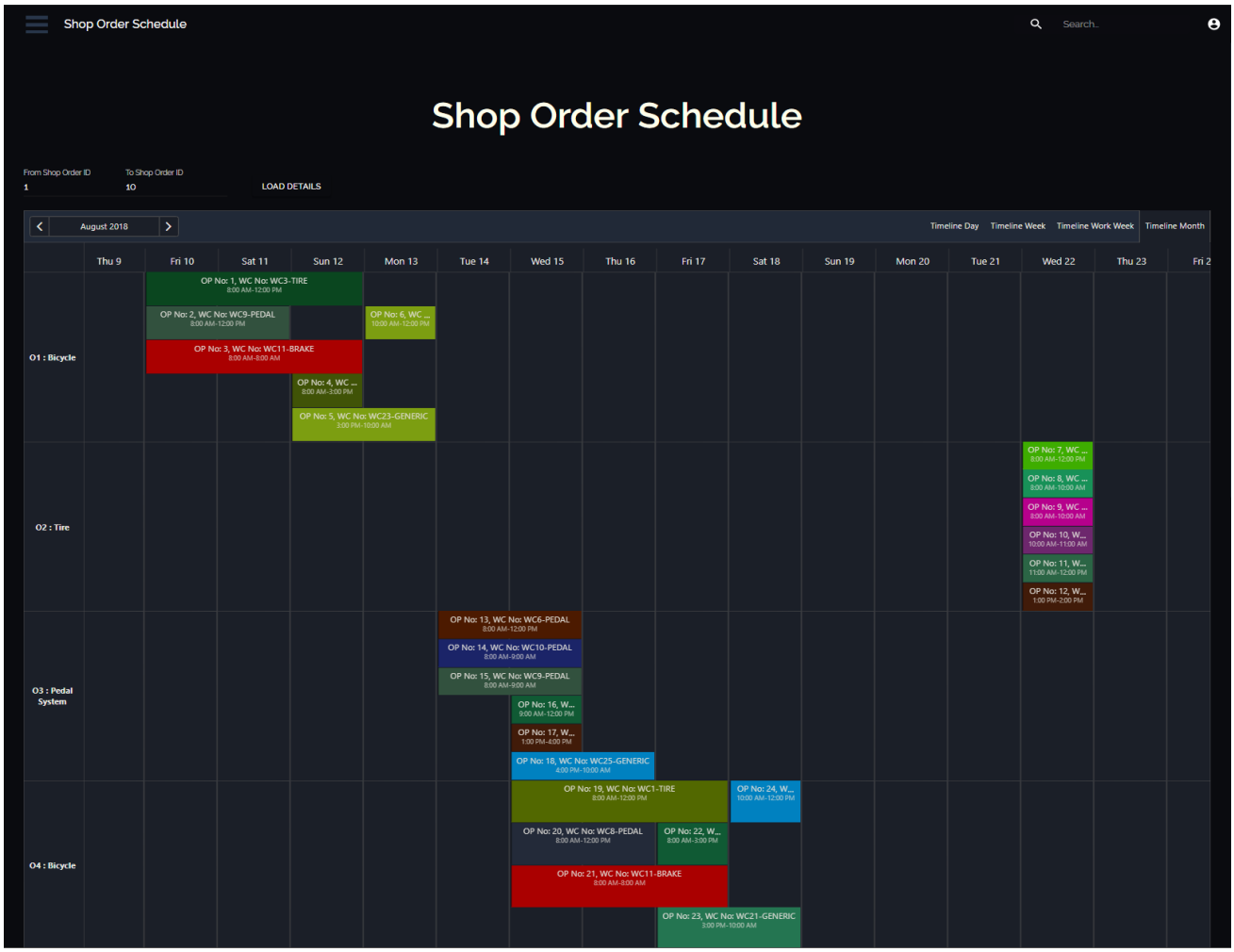


Figure 34.Shop Order Operation Scheduler View

Figure 34 illustrates the current developed Shop Order Schedule UI for the DynoSchedule system.

6.6. Summary

In this chapter the implementation details of the system were discussed in a highly detailed level. There initially, the frameworks and the technologies that were used for the implementation of the system were discussed in terms of the UI, Multi-Agent technology as well as the database level. Afterwards, each of the agent functions were discussed in depth using diagrams such as flowcharts and sequence diagrams. In the next chapter, the evaluation of the proposed system will be discussed with experimental results.

Chapter 7

Evaluation of the DynoSchedule System

7.1. Introduction

This chapter will focus on the evaluation of the developed DynoSchedule System. Per se, the evaluation strategy with the relevant measures and indicators such as work centre availability and Overall Equipment Effectiveness, experimental design with the details of the test dataset, and the experimental results after the test dataset has been integrated and tested with the DynoSchedule system will be discussed in detail.

7.2. Experimental Design

As for the experimental design, the developed DynoSchedule system is compared to an existing constraint-based scheduling system, which utilizes both a manual and dynamic scheduling process, in order to verify the accuracy, effectivity and the efficiency of the DynoSchedule system.

7.2.1. Measurements

When evaluating the developed DynoSchedule system, following listed are the different criteria that will be taken into consideration.

- ***Percentage of orders To Be Completed within the required time:***

  Each order has a Required Date specified by the customer before or on which, the manufactured goods should be delivered. Completing the manufacturing process before reaching this deadline for all the available orders is the ultimate goal of a manufacturing organization and what they try to achieve by using a proper scheduling system. This can be calculated using:

$$\%\ of\ Orders\ TBC\ On\ Time = \frac{No.of\ orders\ TBC\ On\ Time}{Total\ No.of\ orders}\ x\ 100$$

- ***Percentage of tardy orders:***

  This is the no. of orders that cannot meet the Required Date specified by the user (inverse of above). The main purpose of employing a system is to minimize this type of orders as much as possible.

$$\%\ of\ Tardy\ Orders = \frac{No.of\ Tardy\ Orders}{Total\ No.of\ orders}\ x\ 100$$

- ***Utilization of work centres (Availability):***

  Utilization of the work centres is another important indicator that can show the potency of a schedule. A manufacturing organization would ideally like to keep its machineries always running than letting them stay idle, in order to get the maximum output from them, and get a better income as a result. This can be calculated for a specified time period using [26]:

$$WC\ Utilization\ (Availability) = \frac{Avg.Utilized\ Work\ Center\ Hrs}{Avg.Planned\ Work\ Center\ Hrs}\ x\ 100$$

  Average Planned WC Hrs, and Utilized WC Hrs, for ***n*** no. of similar work centres can be calculated using:

$$Avg.Utilized\ Work\ Center\ Hrs. = \frac{\sum_{i=0}^{n}\ Utilized\ Work\ Center\ Hrs}{n}$$

$$Avg.\ \ Planned\ Work\ Center\ Hrs = \frac{\sum_{i=0}^{n}\ Planned\ Work\ Center\ Hrs}{n}$$

  - *Planned Work Centre Hours:* This is the total amount of time for which the work centre is supposed to be producing.

- ***Overall Equipment Effectiveness* (OEE)**:

  OEE is the *golden industry standard* [27] for measuring the productivity of a manufacturing process. Simply put, OEE can be declared as the ratio between the Actual Production Time and the Planned Production Time. Since this provides a bigger picture on the quality of the process, it takes into consideration metrics such as the quality, and the performance as well. However, in terms of using this indicator for the DynoSchedule system evaluation process, those additional indicators are disregarded by setting them to 1. Following displayed is the formula to calculate the OEE of the manufacturing process [26]:

$$OEE = \frac{Good\ count\ x\ Ideal\ cycle\ time}{Planned\ Production\ Time}$$

$$OEE = \ Availability\ x\ Performance\ x\ Quality$$

- *Availability*: Utilization of Work Centres. Calculated by the previous metric.

- *Performance*: calculated using the following formula

$$Performance = \frac{Ideal\ cycle\ time\ x\ Total\ Count}{Run\ time}$$

  - *Ideal Cycle Time*: Theoretical minimum time to produce the part given that it is done continuously without any *Stop Time.*
  - *Stop Time:* The times where the manufacturing process was planned to be running yet were not due to interruptions, both planned an unplanned.

- *Quality*: For the current evaluation purpose, quality values for both the Dataset and the DynoSchedule are taken as the same. Therefore, this doesn't affect the OEE calculation, hence quality is taken as 1.

The target for manufacturing organizations is to achieve the highest OEE value, which is 1 or come closer to it. Usually 0.85 or higher OEE values is considered world-class for a manufacturing organization.

- ***Time taken for the dynamic scheduling process:***

  Time taken for the dynamic scheduling process is another important indicator that should be considered when comparing the DynoSchedule system to an existing adaptive scheduling system algorithm.

7.2.2. System considered for Evaluation

The system that is used for the evaluation purpose of the DynoSchedule system is a conventional ERP Application with features for manufacturing scheduling. There, the system handles dynamic scheduling aspect in 2 main ways:

- *Dynamic Scheduling based on a Machine Learning Approach*: The system utilizes a Machine Learning approach for the dynamic scheduling if and when there are is a disruptive occurrence. However, the users consider this evaluation only when the

disruption is of large scale due to the higher amount of time taken for the process, and the rapid outdate of the entire schedule.

- *Manual Approach*: In simpler cases, the users of this system stick to a manual process where the Shop floor supervisor will be taking control of the scheduling of the operations and change the schedule manually.

7.2.3. Data Set

In order to evaluate the DynoSchedule system, as mentioned, a test dataset has been taken from a manufacturing organization and modified it to fit the data model of the DynoSchedule system. However, the duration values have been set as the same with different order, operation and part names. Table 2 lists the details of the dataset used to evaluate the DynoSchedule system:

Dataset Details	
No. of Orders	80
No. of Operations	500
No. of Work Centres	25
No. of Unique Parts	20
No. of Used parts	14
Time period	06/08/2018 – 31/12/2018
Total Planned Downtime	200hrs
Total Emergency Downtime	22.5hrs (TireWC-6hrs / BrakeWC-12.5hrs / PedalWC-4hrs)

Table 2.Test Dataset Details

7.3. Evaluation Strategy

For the evaluation of the DynoSchedule system, the modified dataset will be imported to the DynoSchedule system database and will run the initial scheduling process. Afterwards, the planned downtimes and the emergency downtimes will be introduced to the system so as to identify how the system behaves and dynamically reschedule the affected shop orders using

the Prioritized Adaptive Scheduling algorithm. Afterwards, the conventional system (with some dynamic scheduling aspects) will be compared to the DynoSchedule system using the previously introduced metrics.

7.4. Experimental Results

Above dataset was replicated in the DynoSchedule system and the initial scheduling algorithm was executed. Afterwards, the planned downtime of 200hrs (8hrs * 25WC; carried out every 3 months) and emergency downtime of 20hrs were introduced to the schedule and allowed the system to dynamically adjust and come into equilibrium. Figure 35 displays a screenshot taken from the DynoSchedule system after the 500 operations have been initially scheduled in different work centres. (The screenshots of this process will be available in more detail in *Appendix A).*

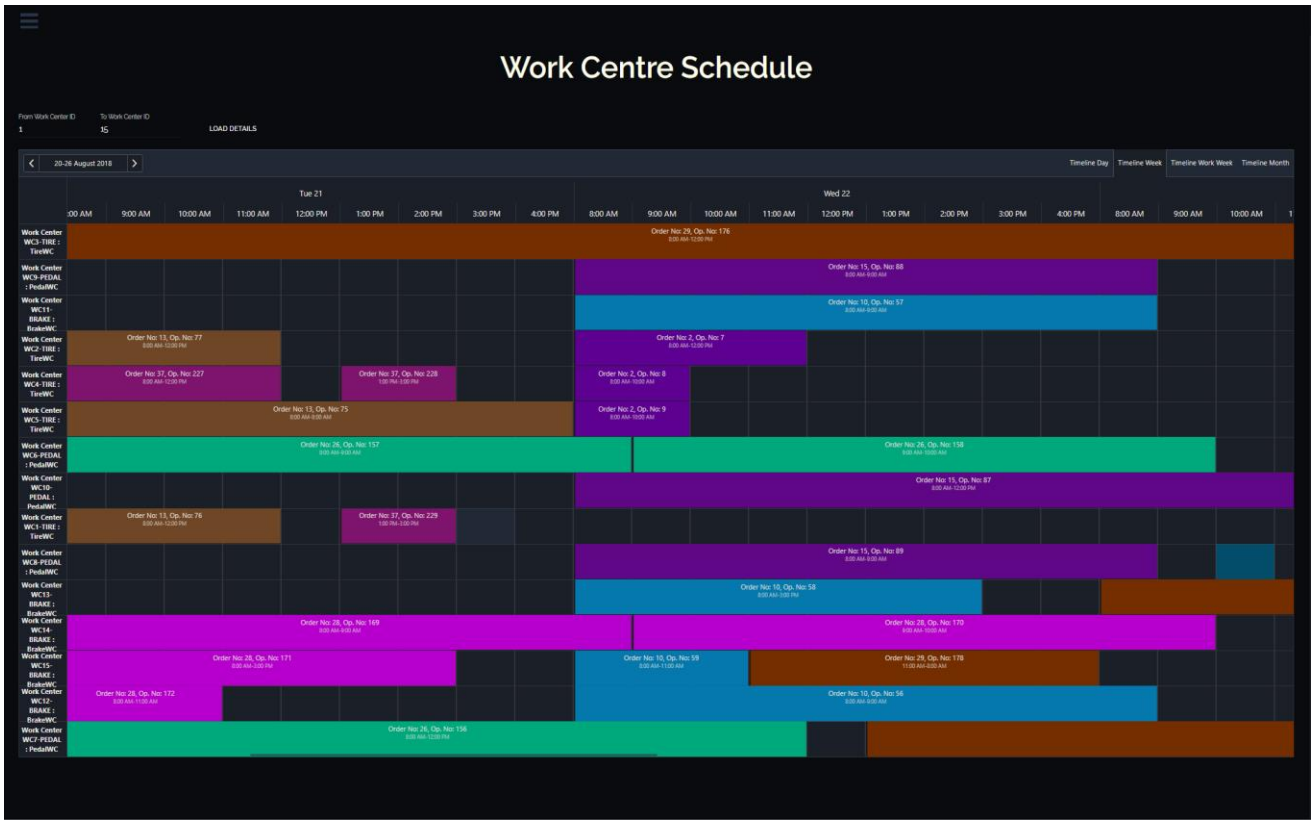


Figure 35.Work Centre Schedule After Initial Scheduling

After introducing the planned and unplanned work centre interruptions to the DynoSchedule system, following are the main outputs taken from it in comparison to the dataset used for the evaluation.

Outputs	Test Dataset	DynoSchedule System
No. of orders TBC on-time	72	80
No. of tardy orders	8	0
Avg. Utilized TireWC Hrs.	143.6Hrs	150.8Hrs (160 - 9.2Hrs)
Avg. Planned TireWC Hrs.	160Hrs	160Hrs (800Hrs / 5)
Avg. Utilized PedalWC Hrs.	84.25Hrs	91.2Hrs (100 - 8.8Hrs)
Avg. Planned PedalWC Hrs.	100Hrs	100Hrs (500Hrs / 5)
Avg. Utilized BrakeWC Hrs.	112.6Hrs	124.5Hrs (135 - 10.5Hrs)
Avg. Planned BrakeWC Hrs.	135Hrs	135Hrs (675Hrs / 5)
Good Count	1	1
Ideal Cycle Time: Tire	8Hrs	8Hrs
Ideal Cycle Time: Pedal	10Hrs	10Hrs
Ideal Cycle Time: Brake	9Hrs	9Hrs
Expected Tire Count Per WC	20	20
Expected Pedal Count Per WC	10	10
Expected Brake Count Per WC	15	15
Actual Tire Count Per WC	16.95 (17.95 – 1)	17.85 (18.85 – 1)
Actual Pedal Count Per WC	8.025 (8.425 -0.4)	8.72 (9.12 – 0.4)
Actual Brake Count Per WC	11.91 (12.51-0.6)	13.23 (13.83-0.6)

Table 3.Outputs Derived from DynoSchedule System compared to Test Dataset

- The planned and the utilized Work Centre hours are calculated for a specified time period, therefore when the Work Centre is affected by interruptions either planned or unplanned, the actual WC hours within that specific time will be lower than the planned WC hours.
- The reason for utilized work centre hours of the test-dataset to be lower than the DynoSchedule system is due to the schedule being extended by interruptions, for more than required time due to inefficiencies in manual scheduling or different

dynamic scheduling algorithms. This exemplifies the effectiveness of the DynoSchedule system with its Prioritized-Adaptive Scheduling algorithm.

- The main reason for the actual Tire, Pedal and Brake counts to have decimals is due to the fact that the data is being normalized for 5 work centres for better comparison and provides the idea of *partial reporting* and *partial completion* of a given part.
- In addition, the reduced amount from the Actual Part counts indicates the damaged goods or the goods that are not up to the expected standard. Since DynoSchedule system relies on estimations, the values for damaged goods available in the test dataset are taken for DynoSchedule system estimation as well. Again, these values can be partial values due to them being normalized.

Using the above outputs, the previously discussed indicators can be calculated for the DynoSchedule System as available on Table 4:

DynoSchedule System Evaluation Measurements			
	TireWC	**PedalWC**	**BrakeWC**
% of Orders TBC On-Time	100%		
% of Tardy Orders	0%		
WC Utilization (Availability)	94.25%	91.2%	92.2%
Performance	94.96%	95.61%	95.63%
OEE	**0.8949**	**0.8719**	**0.8817**

Table 4.DynoSchedule System evaluation measurements

Table 5 illustrates the evaluation metrics calculated for the test dataset.

Test Evaluation Measurements			
	TireWC	**PedalWC**	**BrakeWC**
% of Orders TBC On-Time	90%		
% of Tardy Orders	10%		
WC Utilization (Availability)	89.75%	84.25%	83.40%
Performance	94.42%	95.25%	95.19%
OEE	**0.8474**	**0.8024**	**0.7938**

Table 5.Test Dataset evaluation measurements

Displayed below are the above results in a graph.

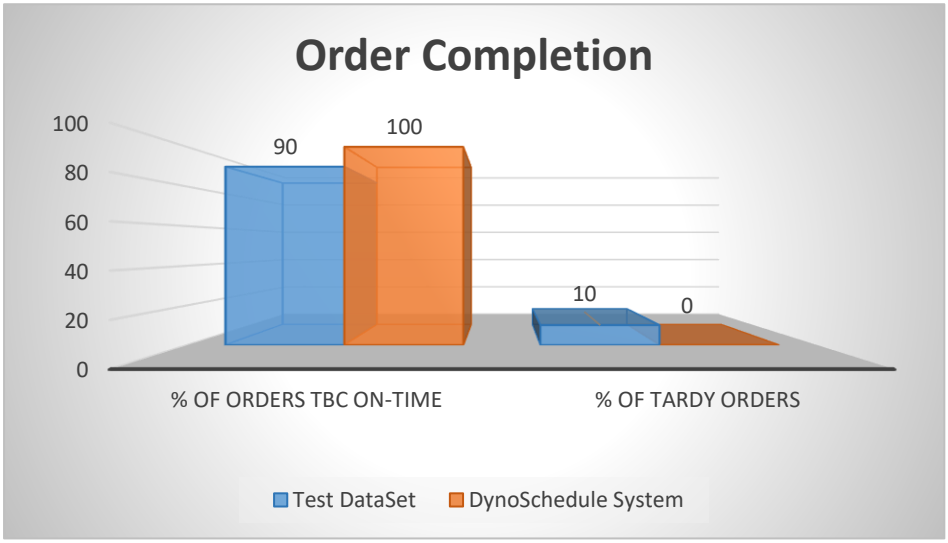


Figure 36.Order Completion Comparison

Figure 36 displays the comparison of the orders to-be-completed and the tardy orders between the test dataset and the DynoSchedule system in a graphical format.

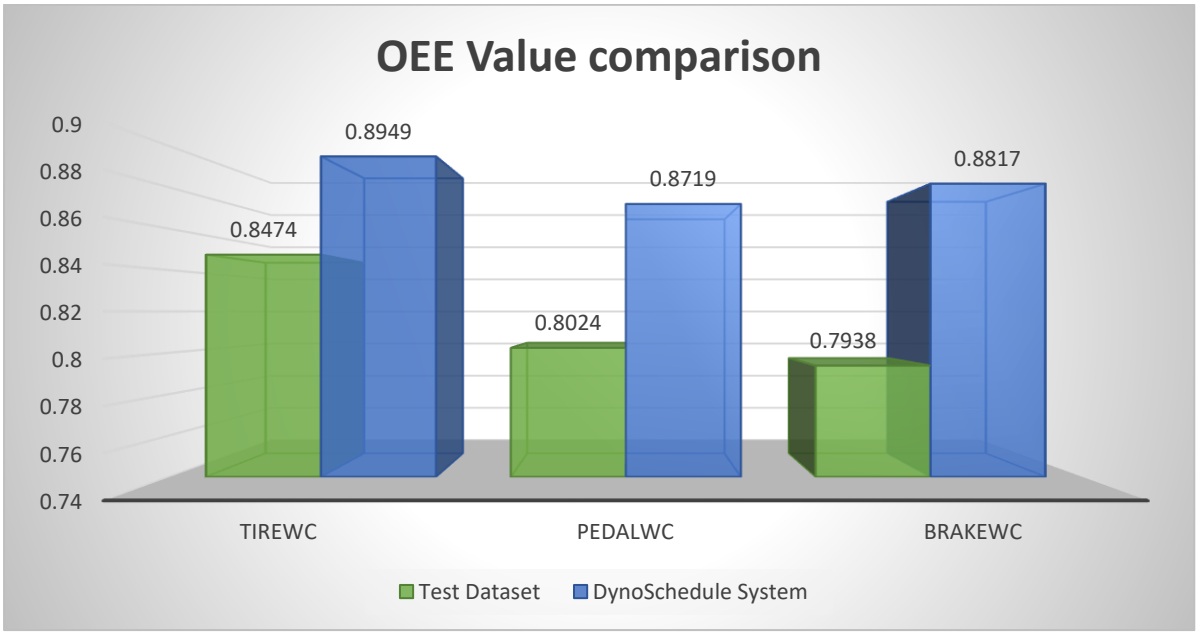


Figure 37.OEE Value comparison

Figure 37 displays the OEE value comparison for different work centres when using DynoSchedule system as well as the test dataset. When contrasting and evaluating the results, it's quite evident how, using the DynoSchedule system provides much better result in terms of

the ability to deliver orders on time as well as the effectivity of work centres, as exemplified by the Overall Equipment Effectiveness calculation.

When it comes to the time taken for the executing of the Dynamic Scheduling process, following graph shows a rough comparison between the times when 1 Shop Order is affected by an interruption:

	Test Dataset	**DynoSchedule**
Total Dynamic Schedule Process Time	25.2s (18 operations rescheduled)	12s (6 operations rescheduled)
Time taken per Operation	1.4s	2s

Table 6.Time taken for Dynamic scheduling

As visible on the table 6, the total amount of time taken for the DynoSchedule system is comparatively lower, yet the amount of time taken per operation is higher than the system from which the dataset is extracted.

7.5. Summary

In this chapter it was discussed how the developed DynoSchedule system was evaluated thoroughly for its functionality as well as the efficiency that it provides for potential manufacturing organizations. There, the experimental results were detailed using tables and diagrams which indicates clear performance improvements when using the DynoSchedule system. In the next chapter, these results will be discussed in detail and conclusions will be determined in terms of the performance of the system as well this study in general. Finally, the limitations of the developed DynoSchedule system as well as the works available on the pipeline will also be discussed.

Chapter 8

Conclusion and Future Work

8.1. Introduction

In the previous chapter, the developed DynoSchedule system was evaluated using different measures, and this chapter will focus on the interpretation of those results and assess the performance of the DynoSchedule system accordingly. In addition, conclusions will be made on how the different objectives identified on the initial stages of the DynoSchedule system have been achieved throughout the timeline of the project. Afterwards, it will deliberate on the limitations and the future works planned for the developed system.

8.2. Conclusion

This study tries to address the common problem of Dynamic Constraint-based Scheduling in manufacturing organizations using a novel Multi-Agent approach called Prioritized-Adaptive scheduling concept, which is an extension to the existing Adaptive-Scheduling algorithms, alongside an advanced market-like negotiation mechanism between different kinds of Agents available in the manufacturing context. As such, the DynoSchedule system has been developed and discussed in detail in the previous chapters of this thesis, also, it has been evaluated using different indicators such as the percentage of orders completed on-time, percentage of tardy orders, work centre utilization and the OEE values for different work centres.

When comparing the results taken from the evaluation, it's apparent that the DynoSchedule system provides much better overall results when compared to the test dataset extracted from an existing scheduling system, which uses both manual and dynamic processing for scheduling. When considering the most critical factor, the no. of orders that can be completed on or before the required date, as can be seen from the results, 10% of the orders failed to deliver on or before the deadline due to being interrupted by planned or unplanned downtimes. However, the DynoSchedule system was able to overcome this by rescheduling the operation on the same time or on a marginally extended time, on a different work centre, which results in only a minor deviation of the finish date of the shop order given by the initial scheduling of it (Appendix A). This had a great impact for metrics such as the work centre availability, performance, and the OEE calculation as well.

When considering the Overall Equipment Effectiveness (OEE) and the related measurements, again the DynoSchedule system has a greater edge compared to the system from the which the dataset was extracted. There, as was visible in the results, DynoSchedule system performs better in every aspect such as the Work Centre Availability and Performance in all considered work centres. The OEE values for the work centres TireWC, PedalWC and BrakeWC were increased by approximate values of 0.04, 0.07, and 0.09 respectively. This is an invaluable prospect for manufacturing organization, since OEE is considered the golden standard that every such organization is striving to reach in the long term.

When it comes to the time taken for the scheduling process, this comes out bearing mix results for the DynoSchedule system. Due to the Prioritized-Adaptive Scheduling algorithm used in the system, only the lower-priority orders will be considered for dynamic scheduling (a maximum no. of 3 per iteration). This results in the system coming to an equilibrium much sooner than expected with a minimum amount of operations being rescheduled. This results in comparatively lower overall time taken for the system to complete the dynamic scheduling process. However, when comparing the time taken to process an operation, it is still quite high; 2 seconds compared to 1.4 seconds of the test system. This is an area where the DynoSchedule system can improve on in the future.

Overall, the DynoSchedule system provided excellent results when compared to the dataset extracted from the system which utilized a combination of manual and dynamic scheduling for their day to day operations. This can mainly be attributed to the Prioritized-Adaptive Scheduling conception introduced in the DynoSchedule system which makes the dynamic scheduling process much more efficient and effective allowing the system to come into an equilibrium much quickly when a disruptive event occurs.

When it comes to the objectives identified in this study, a critical study on Multi-Agent systems and how it has been utilized in different real-world applications as well as the existing literature on how to use Multi-Agent based approach to solve the scheduling problems including production scheduling is available on the second chapter of this thesis. Moreover, a fully functional DynoSchedule system has been developed providing various functions, of which, the design and implementation details were discussed on the fifth and the sixth chapters of the thesis respectively. Afterwards, a critical evaluation of the developed DynoSchedule system has been conducted and the details related to it is available on the chapter seven.

8.3. Limitations of the System

There are a few limitations identified in the developed DynoSchedule system, which are listed below:

- ***Reduced per-operation performance***: as discussed previously, even though the system takes considerably lower amount of time for the dynamic scheduling process to be executed, when considering the time taken per operation, it takes a higher amount of time than the system from which the test dataset was extracted. This is a limitation of the system that can be addressed in the future.

- ***Issues with parallel operations***: Since the Prioritized-Adaptive scheduling algorithm uses only a date from which the lower priority shop orders' operations (which runs on the affected work centre) should be unscheduled, it can result in multiple parallel operations from the shop order that runs through the mentioned date (in different work centres) to be selected by the algorithm to be dynamically unscheduled, which is quite unnecessary. Even though this will not affect the overall schedule of the shop order, this can be a contributing factor for the performance gap discussed in the previous point.

8.4. Future Work

Following listed are the currently identified future works for the DynoSchedule system:

- ***Further improvement to the performance of the system***: Per-Operation performance of the system has been identified as a certain limitation of the system. As a future work, this will be improved by providing proper caching mechanisms on the database, using better design patterns on the code level etc.

- ***Better Feedback and Actions mechanism:*** Currently the system provides feedbacks about the shop orders that were affected by the dynamic scheduling algorithm to the user. However, it doesn't provide any action for users such as Undo or Cancel. As a future work, improvement to the system will be made to DynoSchedule system with regards to the feedbacks and actions, which provide a better overall experience to the user.

- ***Implementing the system in an ERP system:*** ERP systems are used by manufacturing organizations to manage their inventory information and daily operations. Integrating the DynoSchedule system with an existing ERP solution will allow the system to take the relevant data from the ERP system and perform the various functions implemented in the system.

8.5. Summary

The first chapter of this thesis provided a detailed introduction to this study, highlighting its aims and objectives, the background, the identified problem and the proposed solution. Afterwards, the literature around Multi-Agent systems in general as well as the use of Multi-Agent technologies in scheduling process were elaborated, identifying the functional as well as technical opportunities. Subsequently, the identified technologies were discussed in greater detail in the third chapter. The fourth chapter provided details about the Approach followed in the development of the DynoSchedule system was deliberated highlighting its hypothesis, inputs, outputs, process, features etc. The fifth and the sixth chapters of the thesis went into intricate details about the design and the implementation of the DynoSchedule system respectively. There, the details of the system were discussed in terms of the types of agents, processes and functions using diagrams such as flow charts, sequence diagrams and class diagrams. In the subsequent chapter, it was discussed how the DynoSchedule system was evaluated using a dataset acquired from an existing scheduling system in a manufacturing organization. Finally, in this chapter, the conclusions for the DynoSchedule system were discussed highlighting its pros and cons and the improvements that can be done on it in the future.

References

[1] Weiming Shen, Lihui Wang, and Qi Hao, "Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 36, no. 4, pp. 563–577, Jul. 2006.

[2] J. Santos and A. Madureira, "Proposal of Multi-Agent based Model for Dynamic Scheduling in Manufacturing."

[3] C. Le Pape, "Constraint-Based Scheduling: A Tutorial."

[4] E. Alonso, "From Artificial Intelligence to Multi-Agent Systems: Some Historical and Computational Remarks," *Artif. Intell. Rev.*, vol. 21, no. 1, pp. 3–24, 1998.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd edition. Upper Saddle River: Pearson, 2009.

[6] N. Jennings, R., K. Sycara, and M. Woolridge, "A Roadmap of Agent Research and Development," *Auton. Agents Multi-Agent Syst.*, vol. 1, p. 275306, 1998.

[7] T. P. Bagchi, *Multiobjective Scheduling by Genetic Algorithms*. Boston, MA: Springer US, 1999.

[8] S. Andreev, G. Rzevski, P. Shviekin, P. Skobelev, and I. Yankov, "A Multi-agent Scheduler for Rent-a-Car Companies," *Lect. Notes Comput. Sci.*, pp. 305–314.

[9] G. Rzevski, P. Skobelev, A. Ivaschenko, and A. Glaschenko, "Multi-Agent Real Time Scheduling System for Taxi Companies," *Proc 8th Int Conf …*.

[10] G. Rzevski, J. Himoff, M. Hinton, and P. Skobelev, "Magenta technology multi-agent logistics i-Scheduler for road transportation," *Proc. Fifth …*.

[11] M. Tchikou and A. Ramudhin, "Agent-based Approach for Manufacturing Dynamic Scheduling," *2006 Int. Conf. Serv. Syst. Serv. Manag.*

[12] L. M. Camarinha-Matos and R. Rabelo, "Multi-agent-based agile scheduling," *Robot. Auton. Syst.*

[13] R. J. Rabelo and L. M. Camarinha-Matos, "Negotiation in multi-agent based dynamic scheduling," *Robot. Comput.-Integr. Manuf.*, vol. 11, no. 4, pp. 303–309.

[14] R. van der Krogt, J. Little, K. Pulliam, S. Hanhilammi, and Y. Jin, "Scheduling for Cellular Manufacturing," *Lect. Notes Comput. Sci.*, pp. 105–117.

[15] A. Glockner and J. Pasquale, "Coadaptive behaviour in a simple distributed job scheduling system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 902–907, May 1993.

[16] G. Y.-J. Lin and J. J. Solberg, "Integrated Shop Floor Control Using Autonomous Agents," *IIE Trans.*, vol. 24, no. 3, pp. 57–71, Jul. 1992.

[17] P. Valckenaers *et al.*, “A benchmarking service for the manufacturing control research community,” *J. Intell. Manuf.*, vol. 17, no. 6, pp. 667–679, Dec. 2006.

[18] Y. Demazeau, K. Hallenborg, and A. J Jensen, “Reactive agent mechanisms for scheduling manufacturing processes.”

[19] X. Li, W. Li, L. Gao, C. Zhang, and X. Shao, “Multi-agent based integration of process planning and scheduling,” in *2009 13th International Conference on Computer Supported Cooperative Work in Design*, Santiago, Chile, 2009, pp. 215–220.

[20] R. M. Sundaram and S.-S. Fu, “Process planning and scheduling—a method of integration for productivity improvement,” *Comput. Ind. Eng.*, vol. 15, no. 1, pp. 296–301, Dec. 1988.

[21] C. E. Nugraheni and L. Abednego, “Multi Agent Hyper-Heuristics based framework for production scheduling problem,” in *2016 International Conference on Informatics and Computing (ICIC)*, 2016, pp. 309–313.

[22] K. Kravari and N. Bassiliades, “A Survey of Agent Platforms,” *J. Artif. Soc. Soc. Simul.*, vol. 18, no. 1, p. 11, 2015.

[23] “Jade Site | Java Agent DEvelopment Framework.”.

[24] Admin, “Java Advantages and Disadvantages,” *MindsMapped*, 23-Jul-2015.

[25] K. Dyrr, “The Complete Beginner’s Guide to React,” p. 89.

[26] V. Idustries, “The Fast Guide to OEE,” *Vorne Ind.*, p. 27, 2008.

[27] A. J. De Ron and J. E. Rooda, “OEE and equipment effectiveness: an evaluation,” *Int. J. Prod. Res.*, vol. 44, no. 23, pp. 4987–5003, Dec. 2006.

Bibliography

- G. Rzevski, J. Himoff, M. Hinton, and P. Skobelev, “Magenta technology multi-agent logistics i-Scheduler for road transportation,” *Proc. Fifth …*.

- L. M. Camarinha-Matos and R. Rabelo, “Multi-agent-based agile scheduling,” *Robot. Auton. Syst.*

- D. H. Stamatis, *The OEE Primer: Understanding Overall Equipment Effectiveness, Reliability, and Maintainability*, 0 ed. Productivity Press, 2017.

- W. J. van Hoeve, C. P. Gomes, B. Selman, and M. Lombardi, “Optimal multi-agent scheduling with constraint programming,” in *Proceedings of the National Conference On Artificial Intelligence*, 2007, vol. 22, p. 1813.

Appendix A

Evaluating the DynoSchedule system using the Test Dataset

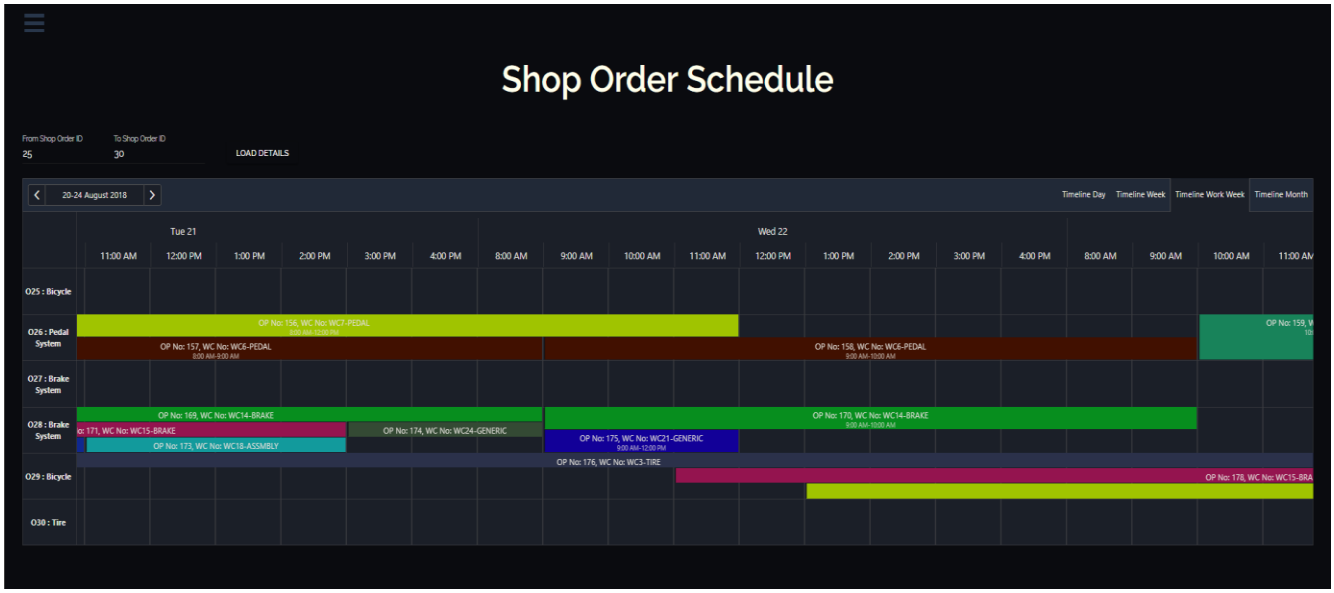


Figure 38. Timeline View after the Initial Schedule – grouped by Shop Orders

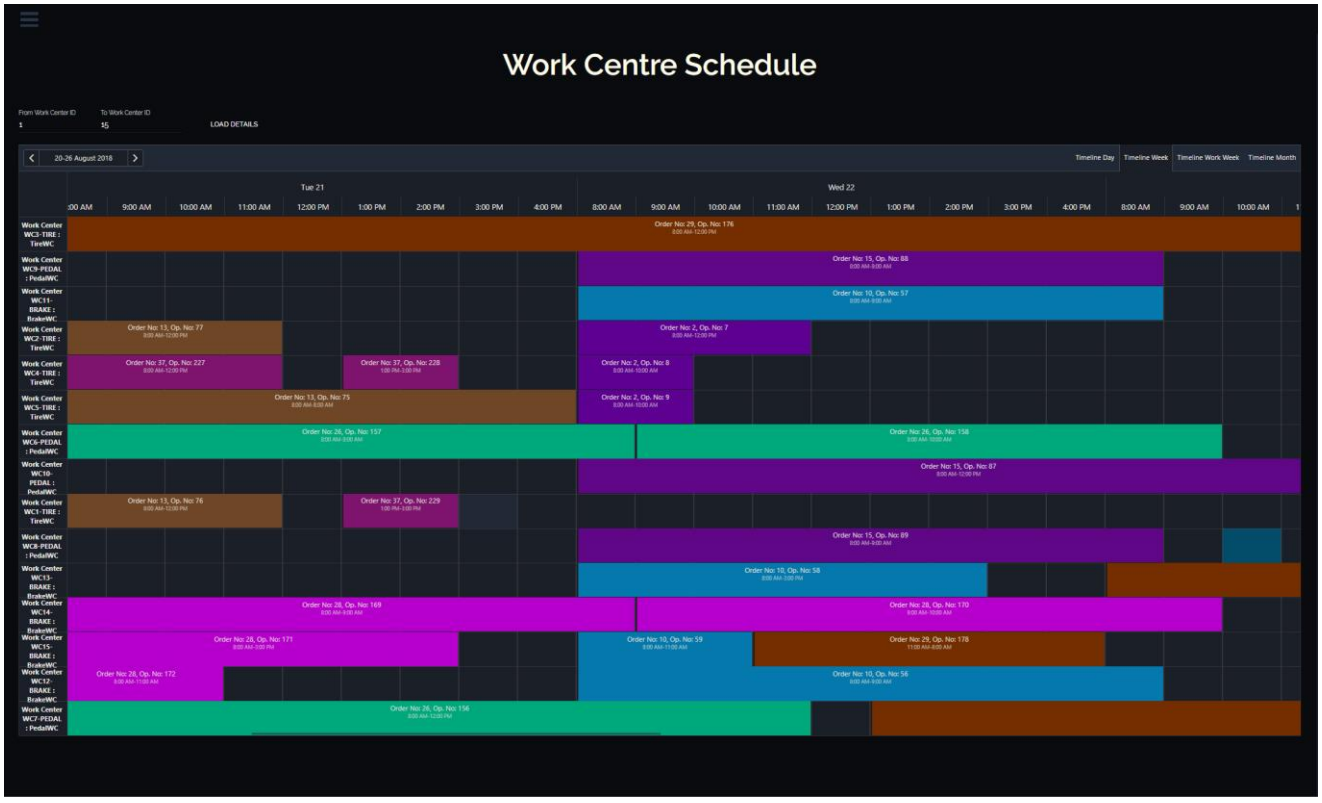
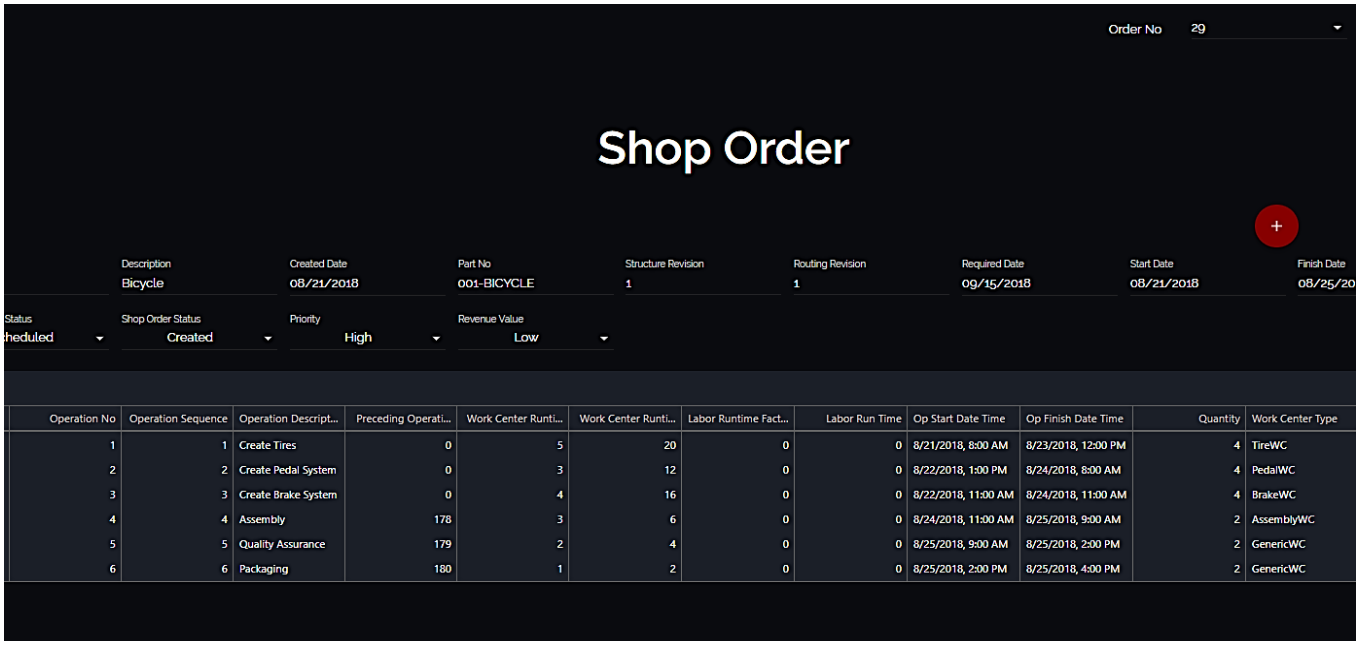


Figure 39.Timeline View after the Initial Schedule – grouped by Work Centres



Operation No	Operation Sequence	Operation Descript...	Preceding Operati...	Work Center Runti...	Work Center Runti...	Labor Runtime Fact...	Labor Run Time	Op Start Date Time	Op Finish Date Time	Quantity	Work Center Type
1	1	Create Tires	0	5	20	0	0	8/21/2018, 8:00 AM	8/23/2018, 12:00 PM	4	TireWC
2	2	Create Pedal System	0	3	12	0	0	8/22/2018, 1:00 PM	8/24/2018, 8:00 AM	4	PedalWC
3	3	Create Brake System	0	4	16	0	0	8/22/2018, 11:00 AM	8/24/2018, 11:00 AM	4	BrakeWC
4	4	Assembly	178	3	6	0	0	8/24/2018, 11:00 AM	8/25/2018, 9:00 AM	2	AssemblyWC
5	5	Quality Assurance	179	2	4	0	0	8/25/2018, 9:00 AM	8/25/2018, 2:00 PM	2	GenericWC
6	6	Packaging	180	1	2	0	0	8/25/2018, 2:00 PM	8/25/2018, 4:00 PM	2	GenericWC

Figure 40.Shop Order No: 29, before interruption

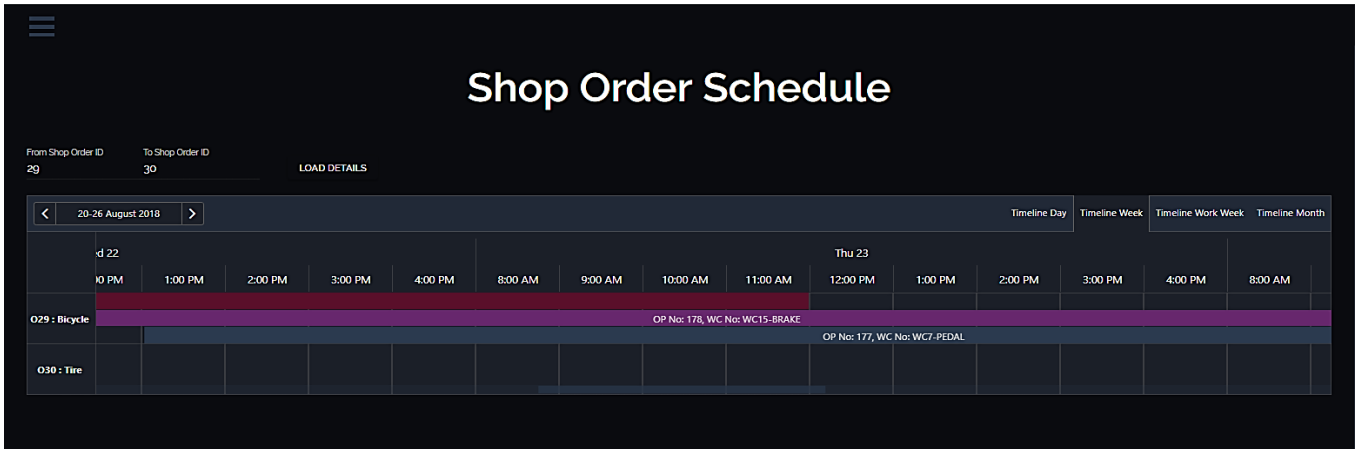


Figure 41.Shop Order 29, before interruption: timeline view

order_no	operation_description	operatio	preced	wc_runtime_fac	wc_runtin	labor_r	op_start_date	op_start_time	op_finish_date	op_finish_time	quantity	work_center_type	work_center_no
29	Create Tires	1	0	5	20	0 0	2018-08-21	08:00:00	2018-08-23	12:00:00	4	TireWC	WC3-TIRE
29	Create Pedal System	2	0	3	12	0 0	2018-08-22	13:00:00	2018-08-24	08:00:00	4	PedalWC	WC7-PEDAL
29	Create Brake System	3	0	4	16	0 0	2018-08-22	11:00:00	2018-08-24	11:00:00	4	BrakeWC	WC15-BRAKE
29	Assembly	4	178	3	6	0 0	2018-08-24	11:00:00	2018-08-25	09:00:00	2	AssemblyWC	WC17-ASSMBLY
29	Quality Assurance	5	179	2	4	0 0	2018-08-25	09:00:00	2018-08-25	14:00:00	2	GenericWC	WC24-GENERIC
29	Packaging	6	180	1	2	0 0	2018-08-25	14:00:00	2018-08-25	16:00:00	2	GenericWC	WC24-GENERIC

Figure 42.Shop Order 29, before interruption: DB view

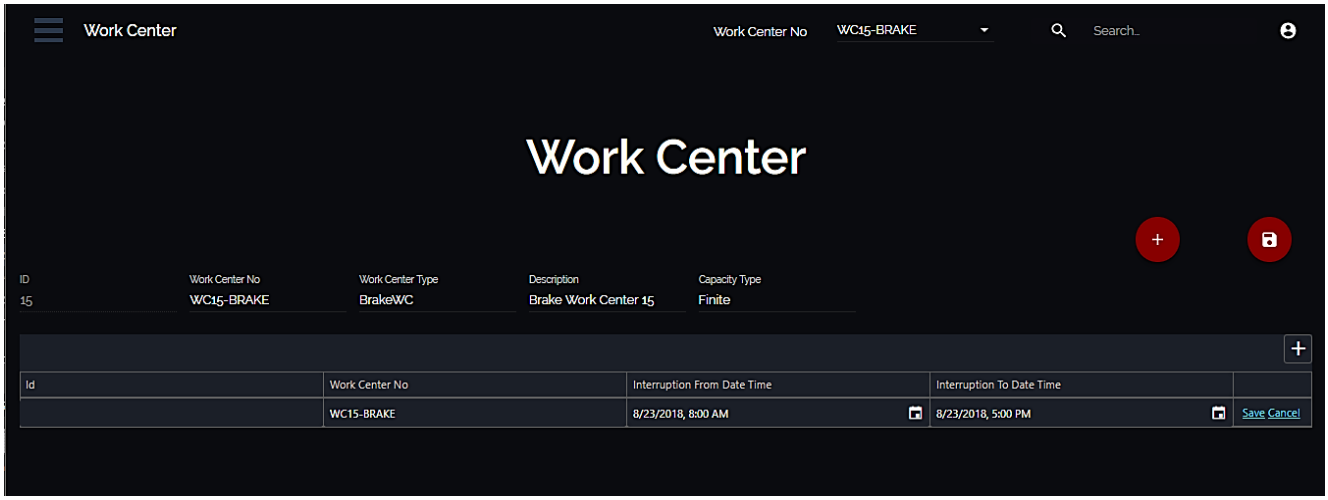


Figure 43.Creating WC Interruption from 23/08/2018 8:00 AM to 5:00 PM

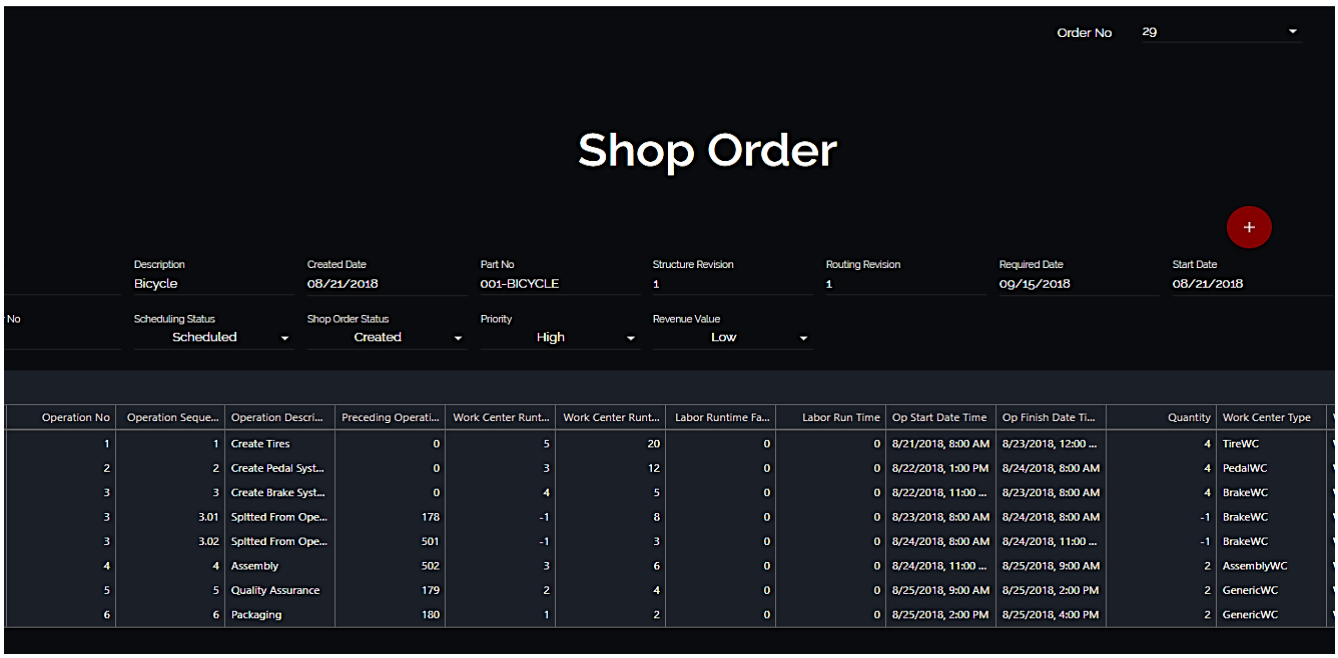


Figure 44.Dynamically Adjusted Shop Order 29

order_no	operation_description	operatio	preced	wc_runtime_fac	wc_runtim	labor_r	op_start_date	op_start_time	op_finish_date	op_finish_time	quantity	work_center_type	work_center_no
29	Create Tires	1	0	5	20	0 0	2018-08-21	08:00:00	2018-08-23	12:00:00	4	TireWC	WC3-TIRE
29	Create Pedal System	2	0	3	12	0 0	2018-08-22	13:00:00	2018-08-24	08:00:00	4	PedalWC	WC7-PEDAL
29	Create Brake System	3	0	4	5	0 0	2018-08-22	11:00:00	2018-08-23	08:00:00	4	BrakeWC	WC15-BRAKE
29	Assembly	4	502	3	6	0 0	2018-08-24	11:00:00	2018-08-25	09:00:00	2	AssemblyWC	WC17-ASSMBLY
29	Quality Assurance	5	179	2	4	0 0	2018-08-25	09:00:00	2018-08-25	14:00:00	2	GenericWC	WC24-GENERIC
29	Packaging	6	180	1	2	0 0	2018-08-25	14:00:00	2018-08-25	16:00:00	2	GenericWC	WC24-GENERIC
29	Splted From Operation ID 178	3.01	178	-1	8	0 0	2018-08-23	08:00:00	2018-08-24	08:00:00	-1	BrakeWC	WC13-BRAKE
29	Splted From Operation ID 178	3.02	501	-1	3	0 0	2018-08-24	08:00:00	2018-08-24	11:00:00	-1	BrakeWC	WC15-BRAKE
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 45.Dynamically Adjusted Shop Order 29: DB View

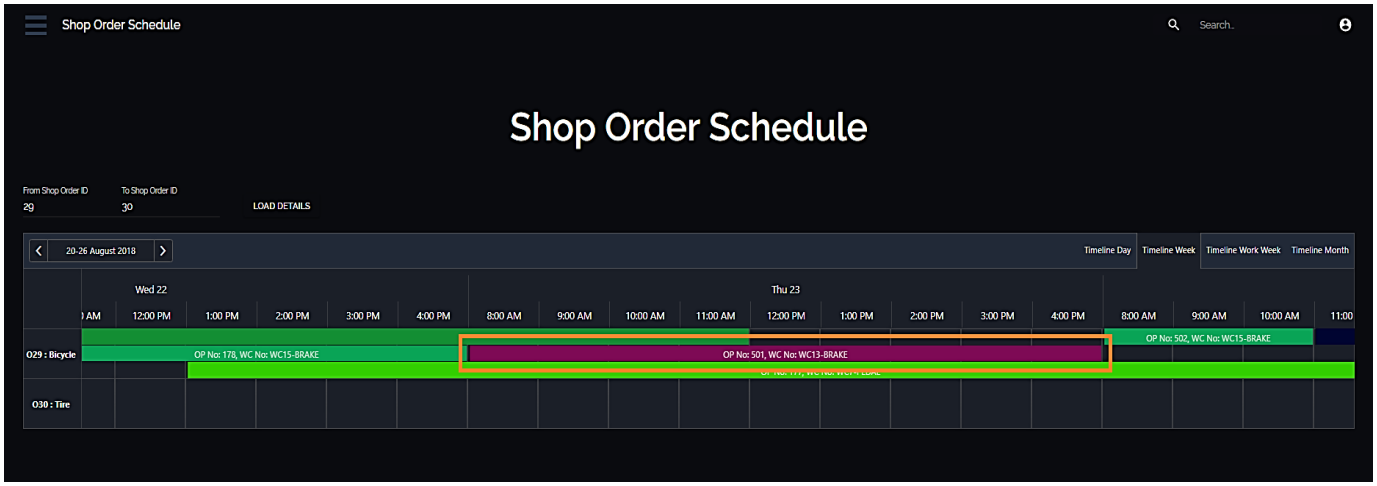


Figure 46.Dynamically Adjusted Shop Order 29: Timeline view

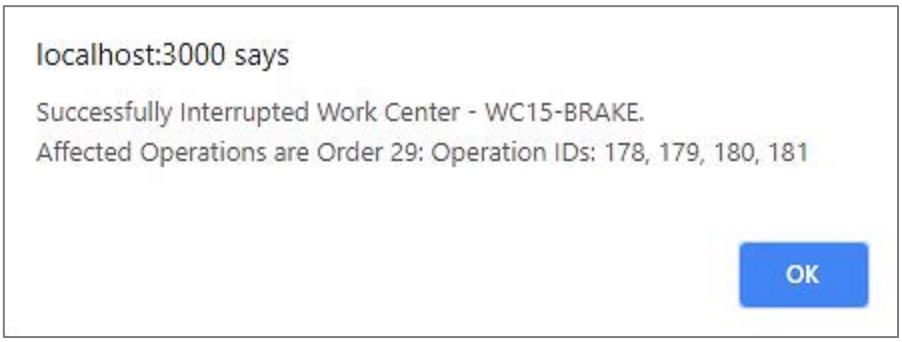


Figure 47.Feedback after Dynamic Adjustment

Appendix B

Code Sections of Different Agent Types, UIs and Message Space Screenshots

```java
/**
 *
 * @author Prabash
 */
public class ShopOrderAgent extends Agent
{

    private static final long serialVersionUID = 884626548653613952 5L;

    // shop order handled by the agent
    private transient ShopOrderModel shopOrder;

    transient DateTimeFormatter dateFormat = DateTimeUtil.getDateFormat();
    transient DateTimeFormatter dateTimeFormat = DateTimeUtil.getDateTimeFormat();

    DateTime unscheduleFromDate = null;
    DateTime unscheduleFromTime = null;

    @Override
    protected void setup()
    {
        super.setup(); //To change body of generated methods, choose Tools | Templates.

        //get the parameters given into the object[]
        final Object[] args = getArguments();
        if (args[0] != null)
        {
            shopOrder = (ShopOrderModel) args[0];

        } else
        {
            System.out.println("Error with the Shop Order arguments");
        }
        addBehaviour(new BQueueNewOperations(shopOrder.getOperations(), shopOrder));
        addBehaviour(new BStartNewOperationScheduler(this));

        System.out.println("the Shop Order Agent " + this.getLocalName() + " is started");
    }

    public ShopOrderOperationModel getOperationById(String operationId)
    {
        ShopOrderOperationModel returnOp = null;
        for (ShopOrderOperationModel operation : shopOrder.getOperations())
        {
            System.out.println(" +++++ operation id : " + operationId);
            System.out.println(" +++++ primary key : " + operation.getPrimaryKey());
            if (operation.getPrimaryKey().equals(operationId))
            {
                returnOp = operation;
                break;
            }

        }
        return returnOp;
        //return shopOrder.getOperations().stream().filter(rec -> rec.getPrimaryKey().equals(oper
    }

    public DateTime targetOpStartDate(String operationId)
    {
        return shopOrder.getOperationTargetStartDate(operationId);
    }
```

Figure 48.Shop Order Agent

```java
/**
 *
 * @author Prabash
 */
public class ManagerAgent extends Agent implements ISchedulerAgent
{

    private static final long serialVersionUID = 3369137004053108334L;
    private static transient List<AgentController> agentList;// agents's ref

    private static final Queue<ACLMessage> NEW_OPERATION_SCHEDULE_REQUESTS_QUEUE = new ConcurrentLinkedQueue<>();
    private static boolean NEW_OPERATION_SCHEDULED;
    private static final Object NEW_OPERATION_SCHEDULE_LOCK = new Object();

    static BQueueNewOperationScheduleRequests queueNewOperationScheduleRequests;
    static BProcessNewOperationScheduleQueue processNewOperationScheduleQueue;
    static BNotifyNewOperationScheduleQueue notifyNewOperationScheduleQueue;
    static BCreateAgents createAgentsBehavior;

    private static final long CREATE_AGENTS_INTERVAL = 30000L;
    private static final long OPERATION_QUEUE_PROCESS_INTERVAL = 5000L;

    @Override
    protected void setup()
    {
        super.setup();

        registerAgentService();

        //get the parameters given into the object[]
        final Object[] args = getArguments();

        if (args[0] != null)
        {
            ContainerController container = (ContainerController) args[0];
            createAgentsBehavior = new BCreateAgents(this, CREATE_AGENTS_INTERVAL, container);
            addBehaviour(createAgentsBehavior);
        }
    }

    @Override
    protected void takeDown()
    {
        super.takeDown(); //To change body of generated methods, choose Tools | Templates.
    }

    @Override
    public void registerAgentService()
    {
        // Register the book-selling service in the yellow pages
        DFAgentDescription dfAgentDesc = new DFAgentDescription();
        dfAgentDesc.setName(getAID());

        ServiceDescription serviceDescription = new ServiceDescription();

        // each agent belonging to a certain work center type will have "work-center-<TYPE>" in here.
        // therefore this should be dynamically set.
        serviceDescription.setType("manager-agent");
        serviceDescription.setName("manager-agent-service");
        dfAgentDesc.addServices(serviceDescription);
        try
        {
            // register the work center agent service
            DFService.register(this, dfAgentDesc);
        } catch (FIPAException fe)
        {
            LogUtil.logSevereErrorMessage(this, fe.getMessage(), fe);
        }
    }
```

Figure 49.Manager Agent

```java
 *
 * @author Prabash
 */
public class WorkCenterAgent extends Agent
{
    private static final long serialVersionUID = 263288753945767774L;
    private transient WorkCenterModel workCenter;

    // the target date requested by the shop order agent for a particular operation
    DateTime requestedOpDate;
    // the best date and timeblock available to the work center agent
    DateTime bestOfferedDate;

    overriden methods

    // <editor-fold desc="behaviors" defaultstate="collapsed">

    // <editor-fold desc="BOfferBestAvailableDate behavior" defaultstate="collapsed">

    /**
     * This behaviour offer the best available date for a specific operation date request
     * sent in by the ShopOrderAgent
     */
    private class BOfferBestAvailableDate extends CyclicBehaviour
    {
        private static final long serialVersionUID = -786010194008349614 8L;

        DateTimeFormatter dateFormat = DateTimeUtil.getDateFormat();
        DateTimeFormatter dateTimeFormat = DateTimeUtil.getDateTimeFormat();

        // <editor-fold desc="overriden methods" defaultstate="collapsed">

        /**
         * action overridden method
         */
        @Override
        public void action()
        {

            MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
            ACLMessage msg = myAgent.receive(mt);
            if (msg != null)
            {
                // CFP Message received. Process it
                String [] messageContent = StringUtil.readMessageContent(msg.getContent());
                requestedOpDate = dateTimeFormat.parseDateTime(messageContent[0]);
                int workCenterRuntime = Double.valueOf(messageContent[1]).intValue();
                String partNo = (messageContent[2]);
                ACLMessage reply = msg.createReply();

                // you should get the date related to the work center that is the earliest date after the target date
                bestOfferedDate = workCenter.getBestDateTimeOffer(requestedOpDate, workCenterRuntime, partNo);

                // reply with the earliest available date/timeblock that comes after the target date
                reply.setPerformative(ACLMessage.PROPOSE);

                // offer should be included with the time as well, therefore the dateTimeFormat is used
                reply.setContent(bestOfferedDate.toString(dateTimeFormat));

                reply.setConversationId("OPERATION_PROCESSING_QUEUE");
                myAgent.send(reply);

            } else
            {
                block();
            }
        }

        // </editor-fold>
    }
```

Figure 50.Work Centre Agent

```java
/**
 * This behaviour gets the acknowledgement for the offered date from the ShopOrderAgent
 * and schedules the operation on the offered date/time
 */
private class BAssignOperationToWorkCenter extends CyclicBehaviour
{
    private static final long serialVersionUID = 466038122618675471 5L;

    // <editor-fold desc="overriden methods" defaultstate="collapsed">

    /**
     * action overridden method
     */
    @Override
    public void action()
    {
        MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
        ACLMessage msg = myAgent.receive(mt);
        if (msg != null)
        {
            // ACCEPT_PROPOSAL Message received with the OperationNo
            String [] messageContent = StringUtil.readMessageContent(msg.getContent());
            String operationId = messageContent[0];
            int workCenterRuntime = Double.valueOf(messageContent[1]).intValue();
            ACLMessage reply = msg.createReply();

            //Integer price = (Integer) catalogue.remove(title);
            if (bestOfferedDate != null)
            {
                reply.setPerformative(ACLMessage.INFORM);

                // return the time block date and the days added
                HashMap<String, Object> timeBlockDetails = workCenter.scheduleOperationFromBestOffer(bestOfferedDate, Integer.parseInt(operationId), workCenterRuntime);
                // set the end time of the operation to be taken as the beginning of the next operation when scheduling
                // in order to do so, increment the received TimeBlockName by 1

                // calculate the finish date time of the current operation (to update on the table/ to be used as the start operation of the next sequential operation)
                LocalDate currentOpFinishDate = bestOfferedDate.plusDays(Integer.parseInt(timeBlockDetails.get(GeneralSettings.getStrDaysAdded()).toString())).toLocalDate();
                LocalTime currentOpFinishTime = WorkCenterOpAllocModel.getTimeBlockValue(timeBlockDetails.get(GeneralSettings.getStrTimeBlockName()).toString());
                DateTime currentOpFinishDateTime = DateTimeUtil.concatenateDateTime(currentOpFinishDate, currentOpFinishTime);

                // set the end date time of the current operation (possible start date of the next operation)
                // and the work center no in the reply content
                reply.setContent(StringUtil.generateMessageContent(currentOpFinishDateTime.toString(DateTimeUtil.getDateTimeFormat()),
                        workCenter.getWorkCenterNo()));

                //update the excel sheet with the date
                System.out.println("WC --> SCHEDULED OPERATION " + Integer.valueOf(operationId) + " ON " + bestOfferedDate);
            }
            reply.setConversationId("OPERATION_PROCESSING_QUEUE");
            myAgent.send(reply);
        } else
        {
            block();
        }
    }

    // </editor-fold>
}
```

Figure 51.Work Centre Agent offer best date behaviour

```
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% NO ORDERS TO BE SCHEDULED!!
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% NO ORDERS TO BE SCHEDULED!!
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% NO ORDERS TO BE SCHEDULED!!
*************** UNSCHEDULING INTERRUPTED OPERATIONS : WC15-BRAKE 2018-08-23T08:00:00.000+05:30 8
*************** UNSCHEDULING INTERRUPTED OPERATIONS : WC15-BRAKE 2018-08-24T08:00:00.000+05:30 3
*************** UNSCHEDULING INTERRUPTED OPERATIONS : WC17-ASSMBLY 2018-08-24T11:00:00.000+05:30 6
*************** UNSCHEDULING INTERRUPTED OPERATIONS : WC24-GENERIC 2018-08-25T09:00:00.000+05:30 4
*************** UNSCHEDULING INTERRUPTED OPERATIONS : WC24-GENERIC 2018-08-25T14:00:00.000+05:30 2
SHOP_ORDER_AGENT29 launched
WORK_CENTER_AGENTWC11-BRAKE launched
WORK_CENTER_AGENTWC12-BRAKE launched
WORK_CENTER_AGENTWC13-BRAKE launched
WORK_CENTER_AGENTWC14-BRAKE launched
WORK_CENTER_AGENTWC15-BRAKE launched
WORK_CENTER_AGENTWC16-ASSMBLY launched
WORK_CENTER_AGENTWC17-ASSMBLY launched
WORK_CENTER_AGENTWC18-ASSMBLY launched
WORK_CENTER_AGENTWC19-ASSMBLY launched
WORK_CENTER_AGENTWC20-ASSMBLY launched
WORK_CENTER_AGENTWC21-GENERIC launched
WORK_CENTER_AGENTWC22-GENERIC launched
WORK_CENTER_AGENTWC23-GENERIC launched
WORK_CENTER_AGENTWC24-GENERIC launched
WORK_CENTER_AGENTWC25-GENERIC launched
Press a key to start the agents

Mar 20, 2019 11:27:40 AM dyno.scheduler.utils.LogUtil logInfoMessage
INFO: Agents started...
the Shop Order Agent SHOP_ORDER_AGENT29 is started
the Work Center agent WORK_CENTER_AGENTWC23-GENERIC is started
the Work Center agent WORK_CENTER_AGENTWC24-GENERIC is started
the Work Center agent WORK_CENTER_AGENTWC16-ASSMBLY is started
the Work Center agent WORK_CENTER_AGENTWC13-BRAKE is started
the Work Center agent WORK_CENTER_AGENTWC17-ASSMBLY is started
the Work Center agent WORK_CENTER_AGENTWC21-GENERIC is started
the Work Center agent WORK_CENTER_AGENTWC11-BRAKE is started
the Work Center agent WORK_CENTER_AGENTWC20-ASSMBLY is started
the Work Center agent WORK_CENTER_AGENTWC19-ASSMBLY is started
the Work Center agent WORK_CENTER_AGENTWC18-ASSMBLY is started
the Work Center agent WORK_CENTER_AGENTWC22-GENERIC is started
the Work Center agent WORK_CENTER_AGENTWC12-BRAKE is started
the Work Center agent WORK_CENTER_AGENTWC25-GENERIC is started
the Work Center agent WORK_CENTER_AGENTWC14-BRAKE is started
ManagerAgent@127.0.0.1:8888/JADE
+++ Operation 29|0.52|501|3.01 from ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))  is
queued!
the Work Center agent WORK_CENTER_AGENTWC15-BRAKE is started
ManagerAgent@127.0.0.1:8888/JADE
+++ Operation 29|0.52|502|3.02 from ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))  is
queued!
ManagerAgent@127.0.0.1:8888/JADE
+++ Operation 29|0.52|179|4.0 from ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))  is
queued!
ManagerAgent@127.0.0.1:8888/JADE
+++ Operation 29|0.52|180|5.0 from ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))  is
queued!
ManagerAgent@127.0.0.1:8888/JADE
+++ Operation 29|0.52|181|6.0 from ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))  is
queued!
operation queue timer thread locked!
 ++++++  Shop Order Agent : ( agent-identifier :name SHOP_ORDER_AGENT29@127.0.0.1:8888/JADE  :addresses (sequence http://DESKTOP-DRAGON:7778/acc ))
 ++++++  operationId : 501
 +++++ operation id : 501
 +++++ primary key : 176
 +++++ operation id : 501
 +++++ primary key : 177
 +++++ operation id : 501
 +++++ primary key : 178
 +++++ operation id : 501
 +++++ primary key : 501
Found the WorkCenterAgents :
```

Figure 52.Screenshot of the Message Space

```java
/*
 * @author Prabash
 */
public class RESTServiceHandler implements Runnable
{

    @Override
    public void run()
    {
        initiateServices();
    }

    public void initiateServices()
    {
        // Create a basic Jetty server object that will listen on port 8080.  Note that if you set this to port 0
        // then a randomly available port will be assigned that you can either look in the logs for the port,
        // or programmatically obtain it for use in test cases.
        Server server = new Server(8080);

        // Create the ResourceHandler. It is the object that will actually handle the request for a given file. It is
        // a Jetty Handler object so it is suitable for chaining with other handlers as you will see in other examples.
        ResourceHandler resource_handler = new ResourceHandler();
        // Configure the ResourceHandler. Setting the resource base indicates where the files should be served out of.
        // In this example it is the current directory but it can be configured to anything that the jvm has access to.
        resource_handler.setDirectoriesListed(true);
        resource_handler.setResourceBase(".");

        //Jersey ServletContextHandler
        ServletContextHandler servletContextHandler = new ServletContextHandler(ServletContextHandler.SESSIONS);
        ServletHolder jerseyServlet = servletContextHandler.addServlet(org.glassfish.jersey.servlet.ServletContainer.class, "/api/*");
        jerseyServlet.setInitOrder(0);
        jerseyServlet.setInitParameter("jersey.config.server.provider.classnames", String.join(",", ShopOrderService.class.getCanonicalName(),
                SampleService.class.getCanonicalName(), WorkCenterService.class.getCanonicalName(), PartService.class.getCanonicalName()));

//        ServletHolder secondServlet = servletContextHandler.addServlet(org.glassfish.jersey.servlet.ServletContainer.class, "/api2/*");
//        secondServlet.setInitOrder(1);
//        secondServlet.setInitParameter("jersey.config.server.provider.classnames",
//                String.join(",", Arrays.asList(SecondEntryPoint.class.getCanonicalName(),
//                        ThirdEntryPoint.class.getCanonicalName())));

        // Add the ResourceHandler to the server.
        HandlerList handlers = new HandlerList();
        handlers.setHandlers(new Handler[]
        {
            resource_handler, servletContextHandler, new DefaultHandler()
        });
        server.setHandler(handlers);

        try
        {
            // Start things up! By using the server.join() the server thread will join with the current thread.
            // See "http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/Thread.html#join()" for more details.
            server.start();
            server.join();

        } catch (Exception ex)
        {
            Logger.getLogger(RESTServiceHandler.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}
```

Figure 53.REST Service Handlers to expose the data to the UI

```
class ShopOrderHeader extends Component {
  constructor(props) {
    super(props);
    this.onShopOrderNoChanged = this.onShopOrderNoChanged.bind(this);
    this.state = {
      shopOrderNos: [],
      shopOrders: [],
      selectedOrderNo: 0,
      currentShopOrder: {}
    };
  }

  componentDidMount() {
    getWCSchedule().then(res => {
      // get the service data
      const serviceData = res.data;
      // send the service data to be formatted
      this.formatShopOrderOperationData(serviceData);
    });
  }

  formatShopOrderOperationData(soOperationData) {
    var currentData;
    if (soOperationData != null) {
      currentData = soOperationData;
    }
    console.log(currentData);

    const shopOrderNos = [];
    const shopOrders = [];
    // Push first default order no.
    shopOrderNos.push({
      value: -1,
      label: "Select Order No"
    });
    for (var i = 0; i < currentData.length; i++) {
      var soObject = currentData[i];
      soObject.createdDate = formatDate(new Date(soObject.createdDate));
      soObject.requiredDate = formatDate(new Date(soObject.requiredDate));
      soObject.startDate = formatDate(new Date(soObject.startDate));
      soObject.finishDate = formatDate(new Date(soObject.finishDate));

      shopOrderNos.push({
        value: soObject.orderNo,
        label: soObject.orderNo
      });
      shopOrders.push(soObject);
    }

    console.log(shopOrderNos);

    this.setState({ shopOrderNos, shopOrders });
  }
```

Figure 54.ShopOrder Header UI code with ReactJs

```
46  class ShopOrderSchedule extends Component {
47    constructor(props) {
48      super(props);
49      this.state = {
50        schedulerHeight : 300,
51        fromShopOrderID : 1,
52        toShopOrderID : 10
53      };
54    }
55
56    componentDidMount() {
57      this.loadShopOrderOperationsBySkipTake();
58    }
59
60    loadShopOrderOperationsBySkipTake(){
61      var skip = this.state.fromShopOrderID - 1;
62      var take = this.state.toShopOrderID - skip;
63
64      getScheduledOrders(skip, take).then(res => {
65        // get the service data
66        const serviceData = res.data;
67        // send the service data to be formatted
68        this.formatShopOrderOperationData(serviceData);
69      });
70
71    }
72
73    formatShopOrderOperationData(soOperationData){
74      var currentData;
75      if (soOperationData == null){
76        currentData = shopOrderOperations;
77      }
78      else{
79        console.log("NOT NULL!")
80        currentData = soOperationData;
81      }
82      console.log(currentData);
83
84      const soOperations = [];
85      const shopOrders = [];
86      const workCenters = [];
87
88      // the height of the scheduler is set dynamically, for each row 400 height is set
89      var height = currentData.length * 150;
90      this.setState({ schedulerHeight : height });
91
92      for (var i = 0; i < currentData.length; i++) {
93        var soObject = currentData[i];
94        let shopOrder = {
95          id: soObject.orderNo,
96          text: "O" +soObject.orderNo + " : " + soObject.description,
97          color: this.getRandomColor()
98        };
99        shopOrders.push(shopOrder);
100       for (var j = 0; j < soObject.operations.length; j++) {
101         var operationObj = soObject.operations[j];
102         operationObj.appointmentHeader = "OP No: " + operationObj.operationId + ", WC N
103         soOperations.push(operationObj);
104
105         let workCenter = {
106           id: operationObj.workCenterNo,
107           text: "Work Center " + operationObj.workCenterNo + " : " + operationObj.workC
108           color: this.getRandomColor()
109         };
```

Figure 55.Shop Order Scheduler UI with ReactJs