ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Melek DEMİRTAŞ

MULTI-AGENT APPROACH FOR THE DYNAMIC SCHEDULING OF MACHINES OF FLEXIBLE MANUFACTURING SYSTEMS

DEPARTMENT OF INDUSTRIAL ENGINEERING

ADANA, 2013

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES

MULTI-AGENT APPROACH FOR THE DYNAMIC SCHEDULING OF MACHINES OF FLEXIBLE MANUFACTURING SYSTEMS

Melek DEMİRTAŞ

MSc THESIS

DEPARTMENT OF INDUSTRIAL ENGINEERING

We certify that the thesis titled above was reviewed and approved for the award of degree of the Master of Science by the board of jury on 06/08/2013.

…….....	……	…….....
Dr. Cenk ŞAHİN	Prof. Dr. Rızvan EROL	Asst. Prof. Dr. Vahit KAPLANOĞLU
SUPERVISOR	MEMBER	MEMBER

This MSc Thesis is written at the Department of Institute of Natural And Applied Sciences of Çukurova University.
Registration Number:

Prof. Dr. Mustafa GÖK
Director
Institute of Natural and Applied Sciences

This study was supported by Scientific Research Project Fund of Ç. Ü.
Project No: MMF2012YL17
This study was supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK).
Project No: 111M279

Not: The usage of the presented specific declarations, tables, figures, and photographs either in this thesis or in any other reference without citation is subject to "The law of Arts and Intellectual Products" number of 5846 of Turkish Republic

ABSTRACT

MSc THESIS

MULTI-AGENT APPROACH FOR THE DYNAMIC SCHEDULING OF MACHINES OF FLEXIBLE MANUFACTURING SYSTEMS

Melek DEMİRTAŞ

ÇUKUROVA UNIVERSITY
INSTITUTE OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF INDUSTRIAL ENGINEERING

Supervisor : Dr. Cenk ŞAHİN
Year:2013, Pages:74
Jury : Dr. Cenk ŞAHİN
: Prof. Dr. Rızvan EROL
: Asst. Prof. Dr. Vahit KAPLANOĞLU

Nowadays, when it comes to increasing the productivity of enterprises, it poses a major importance for them to follow the developments in technology. Under the conditions in which market structure is different and costumers' desires and demands are unsteady, the systems which are more flexible and automated are replaced with traditional manufacturing systems. In manufacturing systems, one of the most effective flexibility types is machine flexibility. When demand fluctuations emerge, machine flexibility has positive contributions as for providing customers' aspirations punctually.

In this study, an agent based system is proposed to simultaneous scheduling of flexible machine groups and material handling system in Flexible Manufacturing Systems. The proposed model is designed by means of Prometheus™ methodology and coded in JACK™ Agent Language. Each agent in the model, which is situated, is autonomous and has an ability to cooperate and negotiate with the other agents in the system. Due to these abilities of agents, the structure of the system becomes more dynamic. The proposed model is tested on test problems from the literature and the results are quite satisfactory. Although the model is designed as online method and has a dynamic structure, the obtained schedules are similar to those obtained from offline heuristic algorithms.

Key Words: Multi-Agent Systems, Flexible Manufacturing Systems, Machine Flexibility

ÖZ

YÜKSEK LİSANS TEZİ

ESNEK ÜRETİM SİSTEMLERİNDE MAKİNELERİN DİNAMİK ÇİZELGELENMESİ İÇİN ÇOKLU ETMEN YAKLAŞIMI

Melek DEMİRTAŞ

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ENDÜSTRİ MÜHENDİSLİĞİ ANABİLİM DALI

Danışman : Öğr. Gör. Dr. Cenk ŞAHİN
Yıl: 2013, Sayfa:74
Jüri : Öğr. Gör. Dr. Cenk ŞAHİN
: Prof. Dr. Rızvan EROL
: Yrd. Doç. Dr. Vahit KAPLANOĞLU

Günümüzde, işletmelerin üretkenliğinin arttırılması söz konusu olduğunda, işletmeler için teknolojideki gelişmeleri takip etmek büyük önem teşkil eder. Pazarın yapısının farklı ve müşteri istek ve taleplerinin değişken olduğu koşullarda klasik üretim sistemlerinin yerini, otomasyona dayalı, daha esnek sistemler almaya başlamıştır. Üretim sistemlerinde birçok esneklik çeşitleri içerisinden en etkili olanlarından birisi de makine esnekliğidir. Makine esnekliği talepteki dalgalanmalar ortaya çıktığı zaman müşterilerin isteklerini zamanında karşılamak adına olumlu katkı sağlamaktadır.

Bu çalışmada, esnek üretim sistemleri içerisindeki esnek makine gruplarının ve malzeme taşıma sistemlerinin eş zamanlı olarak çizelgelenebilmesi için etmen tabanlı bir model önerilmektedir. Önerilen model, Prometheus^{TM} metadolojisi kullanılarak tasarlanmıştır ve JACK^{TM} etmen dilinde kodlanmıştır. Modelde yer alan her bir etmen, kendi kararlarını kendisi alabilecek özerklikte ve sistem içerisinde yer alan diğer etmenlerle işbirliği ve müzakere yapabilecek yeteneğe sahiptirler. Etmenlerin bu özelliklerinden dolayı sistemin yapısı daha dinamik olmaktadır. Önerilen model, literatürden alınan problemler üzerinde test edilmiş ve sonuçlar oldukça tatminkâr bulunmuştur. Model dinamik ve çevirimiçi metot olarak tasarlanmasına rağmen, elde edilen çizelgeler çevirimdışı sezgisel algoritmalardan elde edilen çizelgelere benzerdir.

Anahtar Kelimeler: Çoklu- Etmen Sistemleri, Esnek Üretim Sistemleri, Makine Esnekliği

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Dr. Cenk ŞAHİN who helps for my studies and always shows me the way by guiding ideas. He has given majority confidence to manage to study and a wonderful opportunity to be part of the project team.

I would like to acknowledge the support of my committee members Prof. Dr. Rızvan EROL and Asst. Prof. Dr. Vahit KAPLANOĞLU. They have directed through my study and contribute it by sharing their experiences. I also thank to Prof. Dr. Adil BAYKASOĞLU for his support and contribution.

I am very thankful to Dr. İrfan MACİT for his helps and ideas which conduct to my undergraduate and postgraduate studies.

I would like thankful to Res. Asst. Meral GÜLDEŞ, Res. Asst. Olcay KALAN and Burcu KILIÇARSLAN for their supports.

Giving the great sacrifice for my life, I would like to thank my family. I feel their love all the time.

I also thank to Cukurova University Scientific Research Projects Department for providing financial support during the Master's Degree (Project No: MMF2012YL17) and I would like to thank financial support of scholarships for study project to the The Scientific and Technological Research Council of Turkey (Project No: 111M279).

CONTENTS **PAGE**

ABSTRACT I
ÖZ II
ACKNOWLEDGEMENTS III
CONTENTS IV
LIST OF TABLES VI
LIST OF FIGURES VIII
LIST OF ABBREVIATIONS X
1. INTRODUCTION 1
1.1. Flexible Manufacturing Systems 2
1.2. Flexible Manufacturing Control Systems 4
1.2.1. Centralized Architecture 4
1.2.2. Hierarchical Architecture 5
1.2.3. Heterarchical Architecture 5
1.2.4. Hybrid Architecture 5
1.3. Motivation for the Research 6
2. LITERATURE REVIEW 7
2.1. The Scheduling of Flexible Machines 7
2.2. The Simultaneous Scheduling of Machines and Vehicles 9
2.3. Evaluation of Relevant Literature 13
3. METHODOLOGY 15
3.1. An Overview of Prometheus™ 16
3.2. An Overview of JACK™ 19
4. PROPOSED SYSTEM 23
4.1. Design of Proposed Multi-Agent System 23
4.1.1. System Specifications 23
4.1.2. Architecture Design 26
4.1.3. Detailed Design 28
4.2. Implementation of Proposed Multi-Agent System in JACK™ 38
5. RESEARCH FINDINGS 43

5.1. Test Study 43
6. CONCLUSIONS AND FUTURE WORKS 57
6.1. Summary of the Study 57
6.2. Conclusions 58
6.3. Future Directions 58
REFERENCES 59
BIOGRAPHY 65
APPENDIX 66

LIST OF TABLES **PAGE**

Table 3.1. Key JACK^{TM} Constructs 22
Table 5.1. Travel Time Matrix for the Example Problem.....45
Table 5.2. Data for the Job Sets Used in Example Problems (Part1).....46
Table 5.3. Data for the Job Sets Used in Example Problems (Part2).....47
Table 5.4. Data for the Job Sets Used in Example Problems (Part3).....48
Table 5.5. Comparison of the FMAS with the Offline Methods (for t/p ratio>0.25) (Part1) 49
Table 5.6. Comparison of the FMAS with the Offline Methods (for t/p ratio>0.25) (Part2) 50
Table 5.7. Comparison of the FMAS with the Offline Methods (for t/p ratio<0.25) (Part1) 52
Table 5.8. Comparison of the FMAS with the Offline Methods (for t/p ratio<0.25) (Part2) 53

LIST OF FIGURES **PAGE**

Figure 1.1. Basic Control Architecture of Manufacturing Systems 4
Figure 2.1. The Multi-Agent Architecture.....12
Figure 3.1. The Stages of the Prometheus™ Methodology 17
Figure 3.2. Prometheus™ Software Design.....18
Figure 3.3. Definitions of Protocols in PDT……..……...19
Figure 3.4. A General View of JACK™ Development Environment.....20
Figure 4.1. PDT System Role Definitions……..……..25
Figure 4.2. PDT Agent Types.....26
Figure 4.3. PDT Agent Overview.....27
Figure 4.4. Machine-Operation Agent Negotiation Protocol 28
Figure 4.5. Machine-Machine Resource Agent Negotiation Protocol 28
Figure 4.6. Detailed Design of Manager Agent 29
Figure 4.7. Detailed Design of Staff Agent 30
Figure 4.8. Detailed Design of Operation Agent 31
Figure 4.9. Detailed Design of AGV Agent(Scheduler) ……..……..33
Figure 4.10. Detailed Design of Machine Agent(Scheduler).....35
Figure 4.11. Detailed Design of Machine and AGV Resource Agents 38
Figure 4.12. Create Order Form in the Multi-Agent System 39
Figure 4.13. Staff Agent Architecture and Flow Chart of the CreateOperationAgents Plan of Staff Agent.....40
Figure 4.14. Machine Negotiation Plan of Operation Agent 41
Figure 5.1. The Layout Configurations Used in the Example Problem.....44
Figure 5.2. Comparison between Flexible and Non-Flexible Methods (for t/p ratio > 0.25).....51
Figure 5.3. Comparison between FDE and FMAS (for t/p ratio > 0.25) 51
Figure 5.4. Comparison between Flexible and Non-Flexible Methods (for t/p ratio < 0.25).....54
Figure 5.5. Comparison between FDE and FMAS (for t/p ratio <0.25) 54
Figure 5.6. The GANTT Chart of Including New AGVs 55

LIST OF ABBREVIATIONS

FMS	: The Flexible Manufacturing Systems
CNC	: Computer Numerically Controlled
MAS	: Multi-Agent System
FMAS	: Flexible Multi-Agent System
DE	: Differential Evolution
FDE	: Flexible Differential Evolution
JSSP	: Job Shop Scheduling Problem
FJSSP	: Flexible Job Shop Scheduling Problem
BDI	: Belief-Desire-Intention
UML	: Unified Modeling Language
PDT	: Prometheus™ Design Tool
AGV	: Automated Guided Vehicles
JAL	: JACK™ Agent Language
L/U	: Load/Unload

1. INTRODUCTION

There are some disadvantages of traditional production systems such as a high level of stock of semi-finished goods, long work queues, and low utilization rates. In addition, efficiency and flexibility are the problems which have been on traditional manufacturing systems for a long time. Thus, Flexible Manufacturing Systems have been developed in order to find solutions to these problems. Flexible Manufacturing Systems can be located in almost all the sectors of production.

In the last few decades, new management approaches have been imposed to the operation of manufacturing systems so as to improve the productivity, quality level and cost competiveness of manufacturing systems. The Flexible Manufacturing System (FMS) was demonstrated to meet these challenges of global integration, and its performance is based on the selection of the right scheduling policy as the correct answer (Kumar et al., 2011).

In a manufacturing strategy, one of the important competitive priorities is the flexibility of manufacturing systems (Chang et al., 2001). As being multi-disciplinary, concept flexibility has been defined in various forms by the researchers from different disciplines such as engineering, architecture, biology, economics, and etc. As being an easy and fast way, flexibility can be commented as the ability to respond and adapt to environmental changes. Besides, other expressions for flexibility are adjustable and mobile (Baykasoğlu, 2009; Baykasoğlu and Durmuşoğlu, 2012).

Due to its sophisticated characteristics, researchers especially those who are in the manufacturing area identified many different but related classes of flexibility. Flexibility was defined as eight classes by some researchers, such as machine, process, product, routing, volume, expansion, operation and production flexibility (Browne et al., 1984). Moreover, other researchers mentioned eleven categories of manufacturing flexibility, such as machine, material handling, operation, process, routing, product, volume, expansion, program, production and market flexibility (Sethi and Sethi, 1990).

Flexible Manufacturing System with a conventional scheduler is not able to effectively utilize alternative routes or alternative machine assignments. Fully flexible systems may benefit alternative routes or machine assignments, although it is hard to control of the system (Baykasoğlu and Durmuşoğlu, 2012).

1.1. Flexible Manufacturing Systems

A Flexible Manufacturing System (FMS) is a manufacturing system in which there is a measure of flexibility that allows the system to react in the case of changes, whether predicted or unpredicted. Majority of FMS consist of three main systems; first, the work machines which are often automated CNC machines are connected by a material handling system to optimize parts flow, the second one is the central control computer which controls material movements and the last one is machine flow.

In literature, flexibility is mostly divided into eight different categories or dimensions (Browne et al., 1984);

Machine flexibility: 'the ability to produce a given set of parts types'; for example:

- Set-up times are needed by a machine to pass from one process to another; they include the replacement of the part-programmed, the change of means, the positioning of the part;
- Changing a broken or worn-out means is essential for the time;
- Modifying the set of means on the machine on account of production a different sub-set of parts when time is required;
- There exists required time to set up the new equipment.

Machine flexibility is considered as a basic element allowing a manufacturing system to be flexible. Besides this, it also refers to the various types of operations that the machine can perform without requiring a prohibitive effort in switching from one operation to another (Chang, 2009; Sethi and Sethi, 1990).

Product flexibility: 'the ability to change the process for new part types'. Product flexibility is the most important element in marketing; for example, the rapid start of new products with competitive costs allows an effective response to the market changes. It is measured to one part to another.

Process flexibility: 'the ability to produce a given set of parts types'. The numbers of part types which can be produced are measured of this flexibility. The variations among machine flexibility, product flexibility, and process flexibility are stemmed from machine set-up times, the product mix and the part type processed.

Operation flexibility: 'the ability to interchange the order of operations on parts. According to certain operations, operation of sequencing is too rigid in most cases. Due to the state of the plant, the decisions are taken in real time by the control system.

Routing flexibility: 'the ability to process given set of parts on alternative machines'. A measure of this flexibility provides the number of parts since productivity has decreased. There are two ways to obtain the routing flexibility: the part may be processed in a routing which does not require the use of the machines out of service, or the operations may be performed with other units. This flexibility may be : (a) potential - the processing routings are fixed and only in the case of failures are the alternative ones used; (b) effective - the same part is processed with different routings, independently from failures (De Toni and Tonchia, 1998).

Volume flexibility: 'the ability to operate profitably at varying overall level'. It can be measured by the volume increase/decrease which causes the average costs to reach the maximum acceptable value.

Expansion flexibility: 'the ability to easily add capability and capacity'. This type of flexibility can be determined by the dimensions so that system's capacity can reach such an appropriate flexibilty.

Production flexibility: 'the universe of part types can be processed'. This type of flexibility can also be defined as the potential mix of the parts that can be produced.

1.2. Flexible Manufacturing Control Systems

There are generally four models in control systems, which have been organized respectively: centralized control, hierarchical control, heterarchical control, and hybrid control. Figure 1.1 shows how the control is distributed in each of the models.

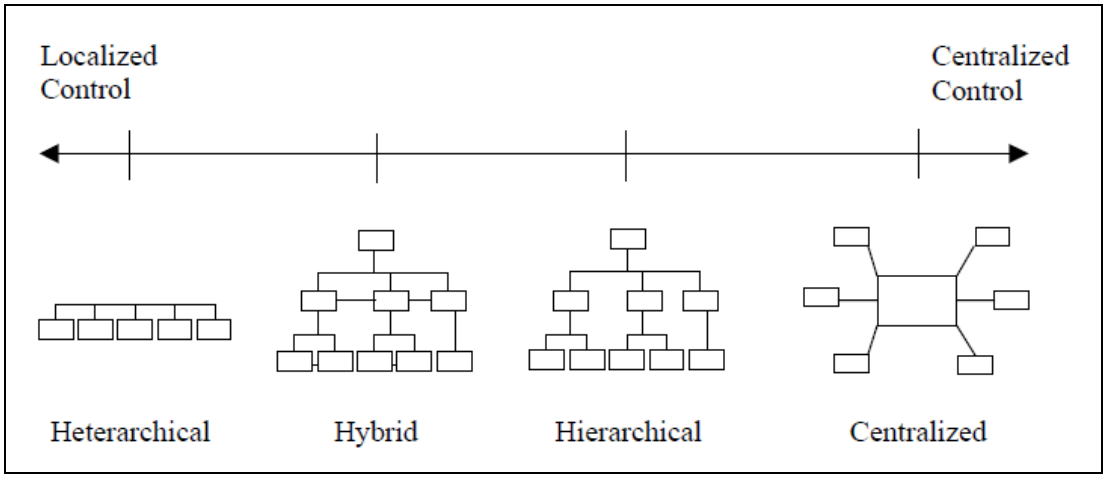


Figure 1.1. Basic Control Architecture of Manufacturing Systems (Duffie et al, 1988)

1.2.1. Centralized Architecture

In the early period of computer controlled automation, centralized control was implemented to the systems. Moreover, all control decisions were made by one central computer, which might have caused some certain difficulties. Such difficulties may have led to certain disadvanteges, one of which is the disadvantage of the centralized architecture for a manufacturing system. Another disadvantage is the control of overall system made by a centralized mechanism, thus limiting the size of the system. The last several decades, the significant increase in available computing power has reduced the severity of this disadvantage (Scott, 2004).

1.2.2. Hierarchical Architecture

To overcome the size limitation of centralized control, hierarchical control was developed for the manufacturing system. The standard business organization is similar to this architecture. Commands are exported by a central authority figure (computer) in the hierarchical control architecture. These commands are interpreted by the next lower level in the hierarchy. Until they reach a level where execution can take place, details are added and the commands are passed to the next lower level in the hierarchy.

1.2.3. Heterarchical Architecture

In the heterarchical control architecture, decisions are distributed. This architecture does not have a central authority. Each machine determines its next operation based on information available locally. No machine knows the complete state of the system, so it is hard to provide global optimization. The advantage of this architecture is not only the reduction in the controller's complexity due to increased modularity and fault tolerance but also reduced software development costs and, developed maintainability and modifiability.

1.2.4. Hybrid Architecture

The hybrid architecture has both the advantages of a hierarchical architecture and heterarchical architecture. Although the system is like heterarchical architecture, a cell control coordinator is added to the architecture. Holonic manufacturing system is an example of this architecture. Holonic manufacturing system provides autonomy ("freedom in decision making") to the individual modules (holons) as it avoids the strict architecture of hierarchical systems, which provides the system with a fast response to disturbances and ability to renew itself in order to face new requirements.

1.3. Motivation for the Research

In recent years, researchers have studied many scheduling problems of manufacturing systems. A number of approaches such as mathematical methods, heuristic methods and simulation methods have been presented for scheduling problems occur in manufacturing systems. Thanks to the advances in computer technology and changes in understanding the business concept, these approaches are becoming obsolete and insufficient to provide proper solutions to problems. Such problems often emerge as a result of the dynamic nature of systems stemming from these changes and improvements.

Agent-based systems have provided an impact for the dynamic scheduling. To re-create and predict the appearance of complex phenomena, the models simulate the simultaneous operations and interactions of multiple agents. Agent-based models consist of rule-based agents which are dynamically interacting. The agents within the systems, in which they interact with, can create real-world-like complexity. Therefore, context of this study is to schedule the machine and vehicles by means of emphasizing flexibility in a flexible manufacturing system through Multi-Agent System (MAS) approach. In particular, it is designed by means of Prometheus™ methodology and then taken a decision for the jobs and transportation with intelligent agents. The stages in the study are as follows;

- Designing a framework for the proposed system's architecture by using Prometheus™ Design Tool.
- Developing a method for machine and vehicles based on MAS in a flexible environment.
- Building (Constructing) a mechanism for interactions and cooperation of the agents in the proposed system.
- Implementing the proposed agent-based system on a real world scheduling problem.
- Benchmarking the proposed system with the others in the literature.

2. LITERATURE REVIEW

There are a number of approaches and proposed solution procedures in terms of the control of vehicle systems and machines in FMS within the scheduling literature, among which computational and manufacturing ones are the most important; therefore, scheduling problems can be understood in general as the problems of allocating resources over time to perform a set of tasks being parts of some processes (Blazewicz et al., 2007).

Research studies are generally field of studies concerned to the simultaneous scheduling of flexible machines and vehicles which are material handling system. In this thesis, the studies divided into two sections which are scheduling of flexible machines, and simultaneous scheduling of machines and vehicles by offline and online methods.

2.1. The Scheduling of Flexible Machines

Among the various types of manufacturing flexibilities that have been addressed in the literature, machine flexibility is the most important and fundamental on which other types of flexibility depend. These types of scheduling problems have only been studied scheduling of machines regardless of vehicles. We can divide the scheduling into two groups: offline and online scheduling just as it is in the previous section; the scheduling of flexible machines.

Genetic algorithm as an offline method was used by Chen et al. (2000) for a class of very complicated combinatorial optimization problems in the job shops. Commonly, the routes of machines visited in these jobs were fixed for flexible job shops, which was not true in the case of flexible manufacturing systems. A new genetic algorithm was presented to solve the flexible job-shop scheduling problem with makespan criterion in their study. Mohamed et al. (2001) studied the relationship between the degree of machine flexibility and the level of system performance. A machine loading and routing model were developed and probed the effect of changing machine flexibility on the performance measures of an FMS. The

results demonstrated that the effect of machine flexibility was more influential than the other factors. A linguistic-based meta-heuristic modeling and solution approach for solving the flexible job shop scheduling problem (FJSSP), which is more complex than the classical job shop scheduling problem (JSSP), in which jobs of varying sizes need to be scheduled on identical machines, was presented by Baykasoglu et al. (2004). In a classical job shop scheduling problem, each job has different routes depending on the restrictions which are fixed and known. Moreover, it has not an optimal solution algorithm. In their study, they modeled a flexible job shop scheduling problem (FJSSP) and solved it thorough widespread usage of numerically controlled multi-purpose machines in modern job shops. Each operation and its orders were assigned to a machine belonging to a set of capable machines (the routing problem). Multiple objective tabu search algorithm was used in the model in their study. Gao et al. (2007) addressed the FJSP problem with three objectives: min makespan, min maximal machine workload and min total workload. They developed a new genetic algorithm hybridized with an innovative local search procedure (bottleneck shifting) for the problem. Wahab et al. (2008) integrated two domains of machine flexibility models which were capability-based machine flexibility and time, cost-based machine flexibility. They proposed a generic model to measure machine flexibility with consideration of uncertainties in the system. Furthermore, part characteristics such as processing time and processing cost, the number of operations that a machine could perform, and indefiniteness in demand and machine-part assignment were included in their approach. Baykasoğlu and Özbakır (2008), effects of process plan and machine flexibilities on the scheduling performance of manufacturing job-shops were analyzed at different flexibility levels by means of the grammar-based flexible job shop scheduling system that was developed. According to the results, the effect of machine flexibility on job shop performance was higher than the process plan flexibility. A new approach which was based on digraph theory and matrix algebra was proposed to quantify flexibility by Baykasoğlu (2009). In the research, the proposed modeling approach was explained through machine flexibility mainly due to the fact that most of the other flexibility types in manufacturing systems rely on this type of flexibility.

Subramaniam et al.(2000) demonstrated that significant improvements to the scheduling performance of dispatching rules could be achieved easily by means of simple machine selection rules. Such rules were proposed in their study and their effectiveness was evaluated through a simulation study of a dynamic job shop. The effects of dispatching rules on the scheduling performance of job-shops with different flexibility levels were analyzed by Baykasoğlu and Özbakır (2010). Four different flexibility levels were defined for operations, and five dispatching rules were deployed according to mean tardiness as the performance criteria for the scheduling system. Vinod and Sridharan (2011) evaluated the performance of the system in terms of various measures based on flow time and tardiness of jobs. The simulation results were subjected to statistical analysis. It was found that dynamic assignment methods provide better performance. Baykasoğlu and Durmuşoğlu (2012) deployed the simulation as an online solution method for their study and analyzed the effects of flexibility on flow time performance of a simulated job shop. As a consequence, several scenarios were developed at four flexibility levels with two different machine selection rules along with three types of dispatching rules. Furthermore, the effect of jockeying as a queuing policy on the flow time performance was also investigated through simulation modeling.

2.2. The Simultaneous Scheduling of Machines and Vehicles

This type of scheduling addresses the problem of concurrent scheduling of machines and a number of identical material handling systems in a Flexible Manufacturing System (FMS). We can divide this type of scheduling into two groups: offline and online scheduling.

To date, in the studies carried out on Flexible Manufacturing Systems (FMSs) and material handling systems, many researchers have preferred offline scheduling methods to find an appropriate schedule. Bilge and Ulusoy (1995) studied the interactions between the machine scheduling and the material handling scheduling in a FMS. They used a nonlinear mixed integer programming model in order to solve machine and material handling system scheduling problem simultaneously. Ulusoy et

al. (1997) presented a genetic algorithm approach to the problem of scheduling of machine and vehicles. A new hybrid genetic algorithm for the simultaneous scheduling problem was investigated by Abdelmaguid et al. (2004) with a view to minimizing the makespan. Reddy and Rao (2006) exploited the simultaneous scheduling problem as a multi-objective scheduling problem which had conflicting objectives. Furthermore, they also used a non-dominating sorting evolutionary algorithm to solve the problem. The study on simultaneous scheduling of machine and vehicles problem was also investigated by Jerald et al. (2006) for a particular type of a FMS environment. They solved the problem by means of a non-traditional optimization technique called the adaptive genetic algorithm. In their study, Subbaiah et al. (2009) addressed the problem of simultaneous scheduling of machines. Moreover, they used two identical automated guided vehicles for material handling systems in a flexible manufacturing system (FMS) to minimize makespan and mean tardiness. In search of finding a proper solution to this problem, a sheep flock heredity algorithm was proposed. The simultaneous optimization of dispatching and conflict-free routing for automated guided vehicles were proposed by Tanaka et al. (2010) in which an application of Petri Net decomposition approach used for dynamic situation where transport requests were given in real time. Babu et al. (2010) also worked the problem of simultaneous scheduling of machines as well as two identical automated guided vehicles in a flexible manufacturing system. In their study, a differential evolution algorithm was proposed to solve the problem.

Kumar et al. (2011) first considered that flexibility of machines by means of offline alghorithm simultaneously. In their study, they addressed the simultaneous scheduling of both machines and material handling system with alternative machines to minimize the makespan by means of offline method. Accordingly, machine flexibility and flexible routing for operations were considered. The authors proposed a machine selection heuristic and a vehicle assignment heuristic which were incorporated in the differential evolution approach to assign the tasks to appropriate machines and vehicles, and thus minimizing cycle time.

Besides offline methods, online methods which can be divided into two sections; as centralized and decentralized are included in the study. One of those

studies including centralized control was Egbelu and Tanchoco (1984). They performed one of the first simulation studies which tested a number of scheduling rules to dispatch vehicles. For machines and vehicles, scheduling rules were studied by Sabuncuoglu and Hommertzheim (1989, 1992, and 1993) in a FMS. Firstly, they investigated the performances of machine and vehicle scheduling rules based on mean flow-time criterion. Simulation method was used to measure the performance of different scheduling rules. Afterwards, they proposed an algorithm with a better sight to compare the scheduling rules. Finally, an experimental investigation of scheduling rules for a wide variety of objective functions was presented by means of a simulation model of FMS.

In order to show how different routing methods perform for a specific design, decentralized control of online scheduling, a multi-agent architecture for the control of material handling system was presented by Farahvash and Boucher (2004) Wallace (2007) proposed a research which was directed into the utilization of heuristics to improve the performance of an agent-based controller. An intelligent agent-based framework was presented by Srivastava et al. (2008) to solve some of the complex problems related to collisions among the vehicles. According to their simulation results, the utilities of their study obtained were an optimal path, less computational burden and higher efficiency. Komma et al. (2011) investigated on modeling of different agents in manufacturing area on JADE reactive architecture. They used JADE for describing vehicles, machine and other agents in their system.

Erol et al. (2012) studied a multi-agent based scheduling approach for machines and vehicles within a flexible manufacturing system which was decentralized control of online scheduling. A newly grown area of distributed artificial intelligence is a multi agent-based system, which provides some effective mechanisms for the management of dynamic operations in manufacturing environments. The model for scheduling of vehicles and machines is shown in Figure 2.1 from the viewpoint of manufacturing system; this architecture consists of a Manager Agent, an AGV System Holon, an Order System Holon and a Machine System Holon. In their study, Holon defined as an agent group.

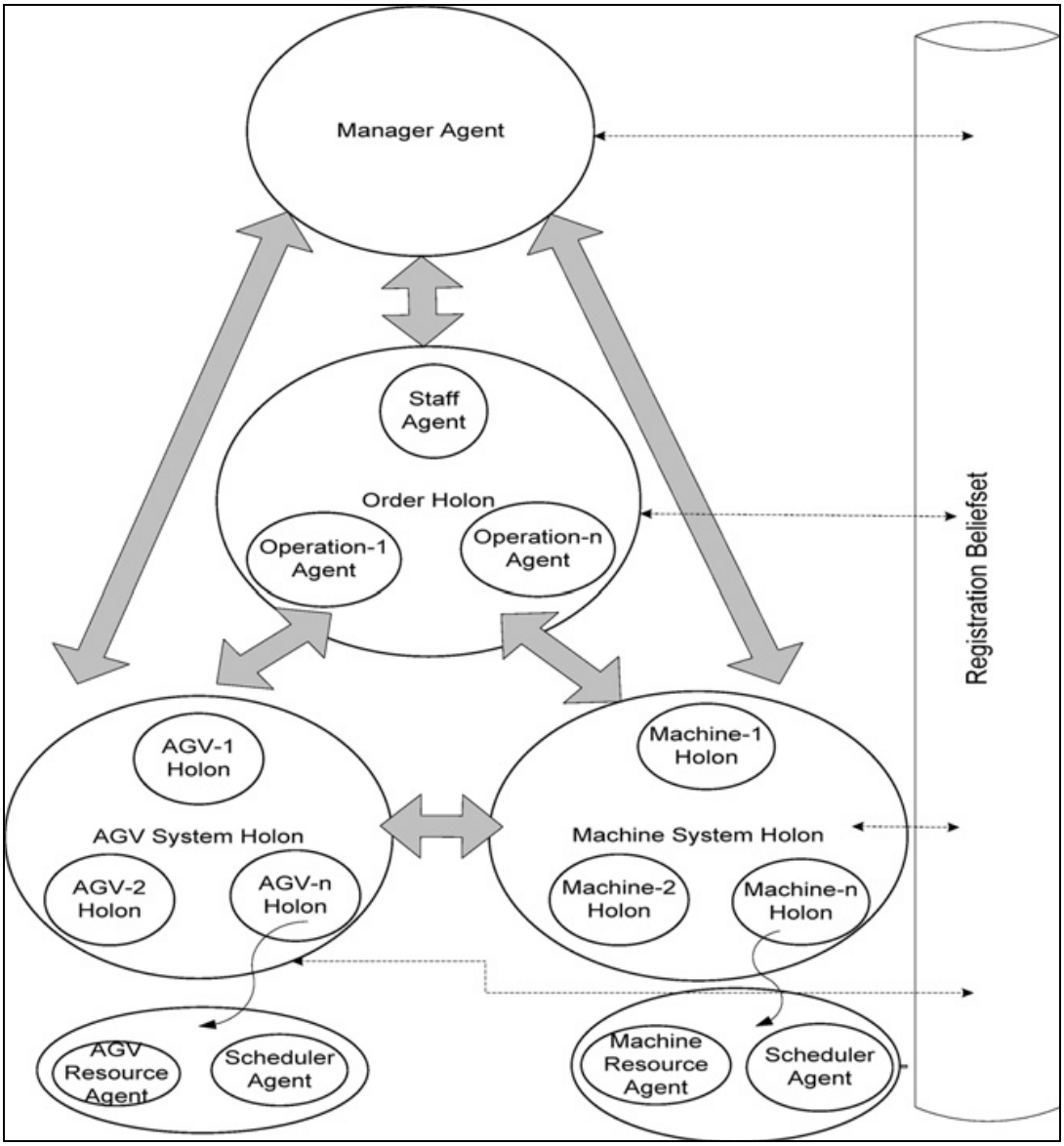


Figure 2.1. The Multi-Agent Architecture (Erol et al., 2012)

The agent types within the agent based design were determined depending on their functions and respective system goals. In their model, two groups of agents were introduced: physical and logical agents. Logical agents were defined as Manager Agent, Staff Agent, Operation Agent Type, AGV Agent Type (Scheduler), Machine Agent Type (Scheduler). Physical agents were defined as AGV Resource Agent Type, Machine Resource Agent Type. According to their approach, the test problem from Bilge and Ulusoy (1995) was chosen which solved as an offline scheduling problem. The approach was used to compare their study results,

accordingly. The performance criterion was selected as makespan minimization in their study. This approach was tested on online and offline scheduling problems from the literature but; however, they did not handle machine flexibility for operations in the system.

2.3. Evaluation of Relevant Literature

After reviewing scheduling in FMS environment, machine scheduling with flexibility, and simultaneous scheduling both machines and vehicles in Flexible Manufacturing Systems, it is found that some key methods can be added to the studies conducted.

Machine flexibility problem in scheduling literature is studied and it is obvious that the number of the studies dealt with offline scheduling are much more than online scheduling studies. Moreover, these studies were performed only by considering scheduling of machines, whereas the scheduling of vehicles were not considered as an issue in these studies. For the proposed system, simultaneous scheduling of machines and vehicles in the Flexible Manufacturing System is presented in addition to scheduling of flexible machines.

Simultaneous scheduling of machines and vehicles studied in the second section of the literature are in line with the current study. It was observed that offline and online scheduling in the researches considered only flexibility of routes, whereas they overlooked machine flexibility. The current study is a synthesis of these approaches integrating machine flexibility with multi agent based system in FMS. In this study, multi agent systems are built for scheduling of machines and vehicles simultaneously in the FMS taking machine flexibility into account.

3. METHODOLOGY

In recent years, there has been a general trend to increase the intelligence of control and reactivity to changes. Agent technology is currently considered as an important approach for developing intelligent manufacturing systems. Agent systems are becoming more popular than the other methods in terms of solving a wide range of complex problems.

A Multi-Agent System is a collection of organized intelligent software agents which work together or compete to find the answer to problems that are complex for single software agents. The only way it can reasonably be addressed is to develop a number of functionally specific and modular components (agents) that are specialized in solving a particular problem appearance if a problem domain is particularly complex, large, or unpredictable (Sycara, 1998).

An agent is defined as a computer system that is situated in some environment and capable of autonomous action in this environment in order to meet its design objectives (Wooldridge and Jennings, 1995; AOS, 2009). According to definition of Farahvash, an agent is an autonomous, computational entity that can be viewed as perceiving its environment and acting upon it (Farahvash and Boucher, 2004). An intelligent agent is a piece of software that has some characteristics which are the following (Evertsz et al., 2004; Padgham and Winikoff, 2004);

- Flexible
- Autonomous
- Reactive
- Proactive
- Situated
- Robust
- Social

BDI (Belief-Desire-Intention) architectures are probably the most popular agent architectures. The BDI paradigm was developed as a response to a perceived problem with existing 'Artificial Intelligence' approaches to the planning problem. The planning problem can be stated informally as: "Given a goal, generate a sequence of actions that will achieve that goal" (Evertsz et al, 2004). In this type of agents, decision making depends on the manipulation of data structures representing beliefs, desires, and intentions of the agent. These agents are useful for systems that need human-like behavior (Komma et al., 2011).

In this study, the proposed architecture is based on an intelligent system. The proposed MAS is designed using Prometheus™ methodology and coded and implemented in JACK™ agent software. In this chapter, the methodology in order to design of agent based system for the scheduling problem domain is introduced.

3.1. An Overview of Prometheus™

When it comes to the relationship between the technology and software development, the key issue is the development of appropriate methodologies for an engineering agent-oriented software. A Multi-Agent System could be designed by utilizing the methodology of Prometheus™, which is a general purpose methodology for the development of software agent systems free from any specific model of the agency of the software platform (Padgham and Winikoff, 2004).

The Prometheus™ design methodology comprises three distinct but interconnected stages. Figure 3.1. represents the stages of modelling the Prometheus™ agent methodology (Padgham and Winikoff, 2004; Baykasoglu et al., 2011):

- System specification stage: at this stage, the system is specified by focusing on the goals and basic functionalities of the system.
- Architectural design stage: at this stage, the agent types are determined according to the system specification stage. It is similar to building class

diagrams from the requirements for documents in the unified modelling language (UML).

- Detailed design stage: this stage includes the plan definition of the agent types.

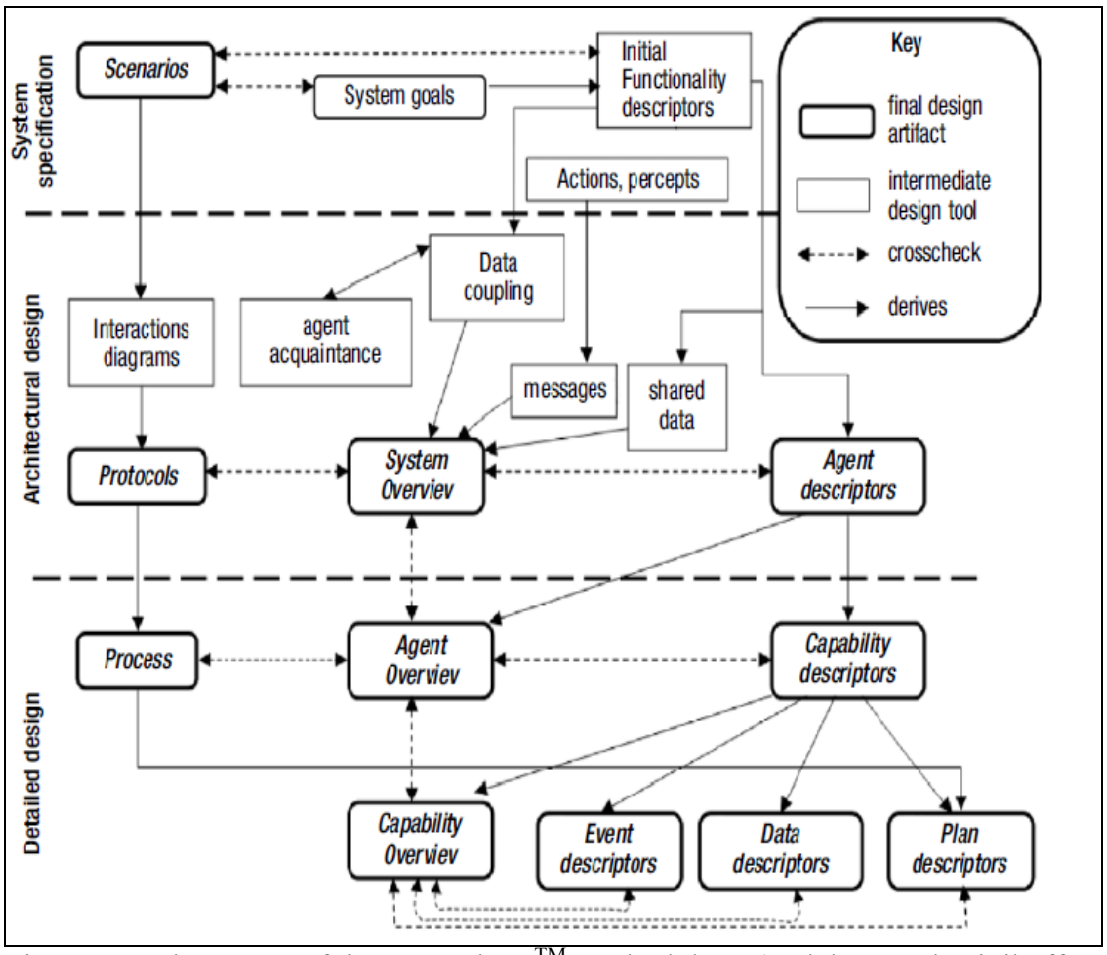


Figure 3.1. The Stages of the Prometheus™ Methodology (Padgham and Winikoff, 2004)

Prometheus™ Design Tool (PDT), one of the most popular design tools, which was developed in conjunction with Prometheus™ method, is used in the design of agent-based systems. This design tool was developed by School of Computer Science and Information Technology at RMIT University. Software can be downloaded available for free of charge via the Internet (http://www.cs.rmit.edu.au/agents/pdt/). A general view of software is seen in Figure 3.2. As shown in Figure 3.2, software consists of three stages.

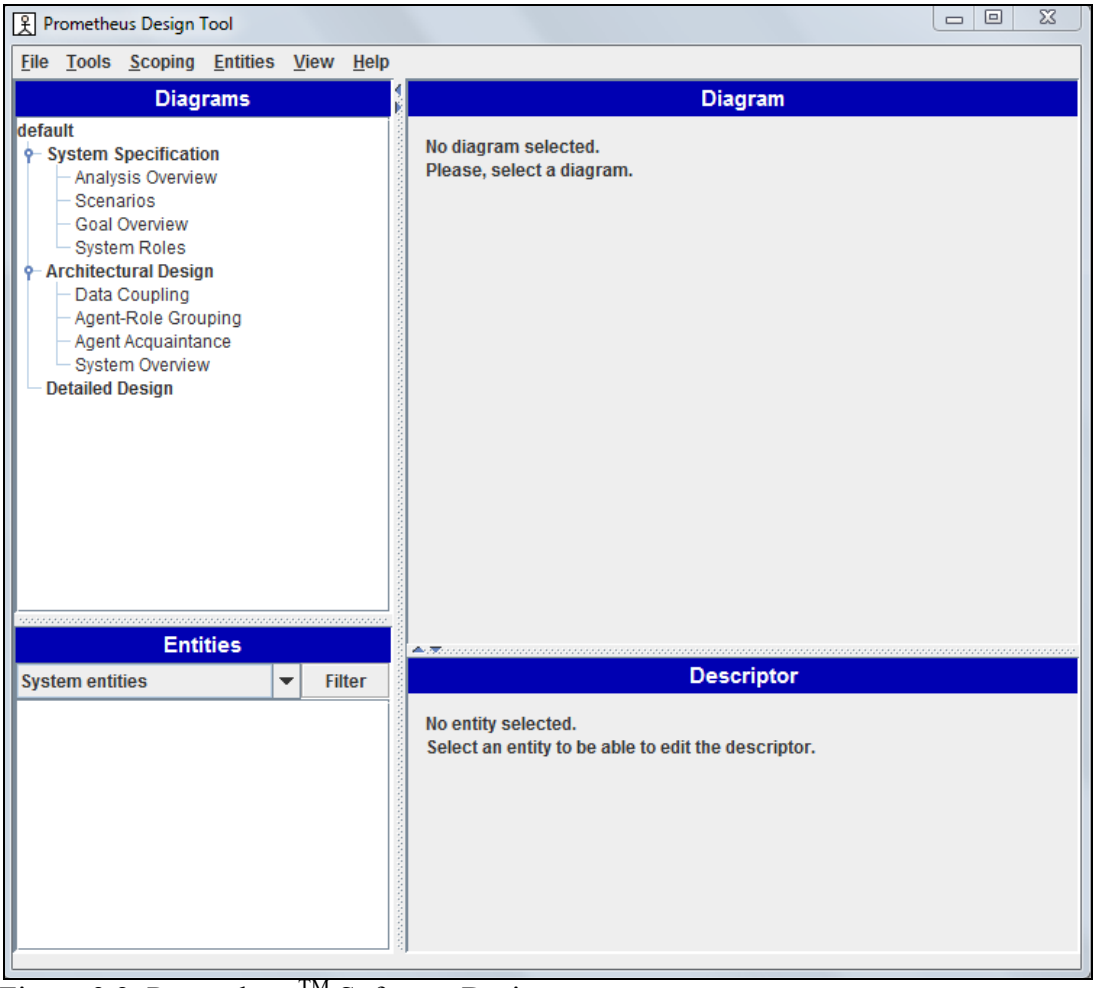


Figure 3.2. Prometheus™ Software Design

PDT software encodes a base in order to implement agent-based systems. The code is generated for agent-based systems by PDT in which coding stages constitute the skeleton of the system. Protocols among agents are identified in the PDT. Code generating in PDT software and imaging properties of the protocol are shown in Figure 3.3.

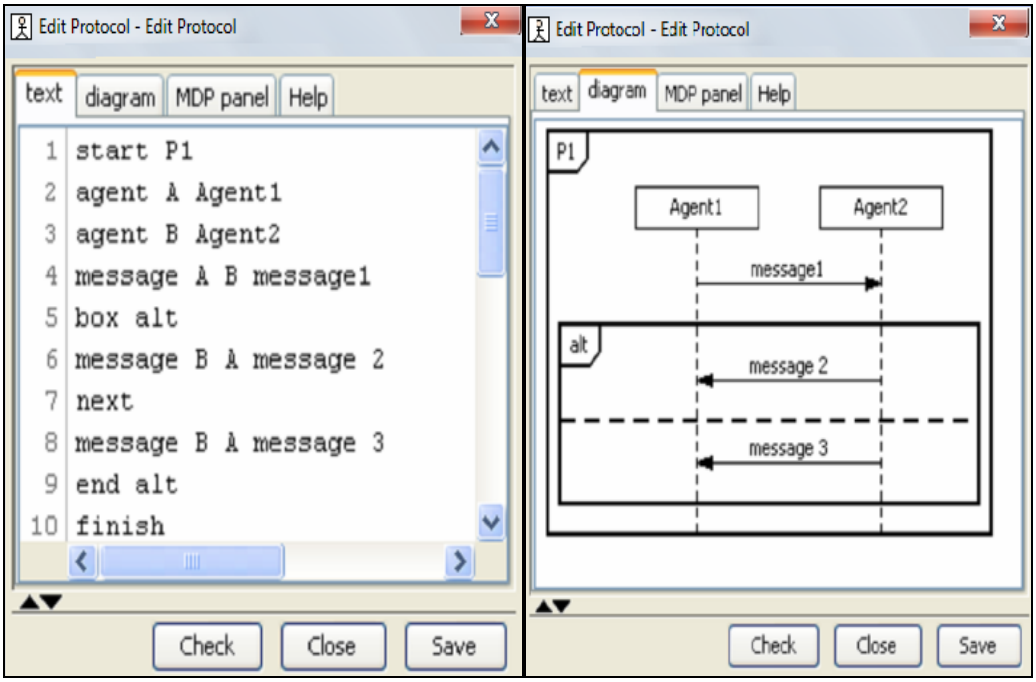


Figure 3.3. Definitions of Protocols in PDT

3.2. An Overview of JACK™

JACK™ is an Agent Oriented development environment which is integrated with the Java™ programming language. Therefore, it is possible to insert standard Java™ codes into any place of JACK™ Agent Language (JAL). This provides a huge flexibility to JACK™. At first JACK™ source code is compiled into regular Java™ code before its being executed.

JACK™ is the world's leading agent infrastructure for building and deploying autonomous systems. It is based on the BDI paradigm (Beliefs/Desires/ Intentions). JACK™ is a mature implementation of the BDI paradigm, written as an extension of Java™ (Figure 3.4.). It has a sophisticated graphical development environment that supports the design, implementation as well as monitoring BDI agents.

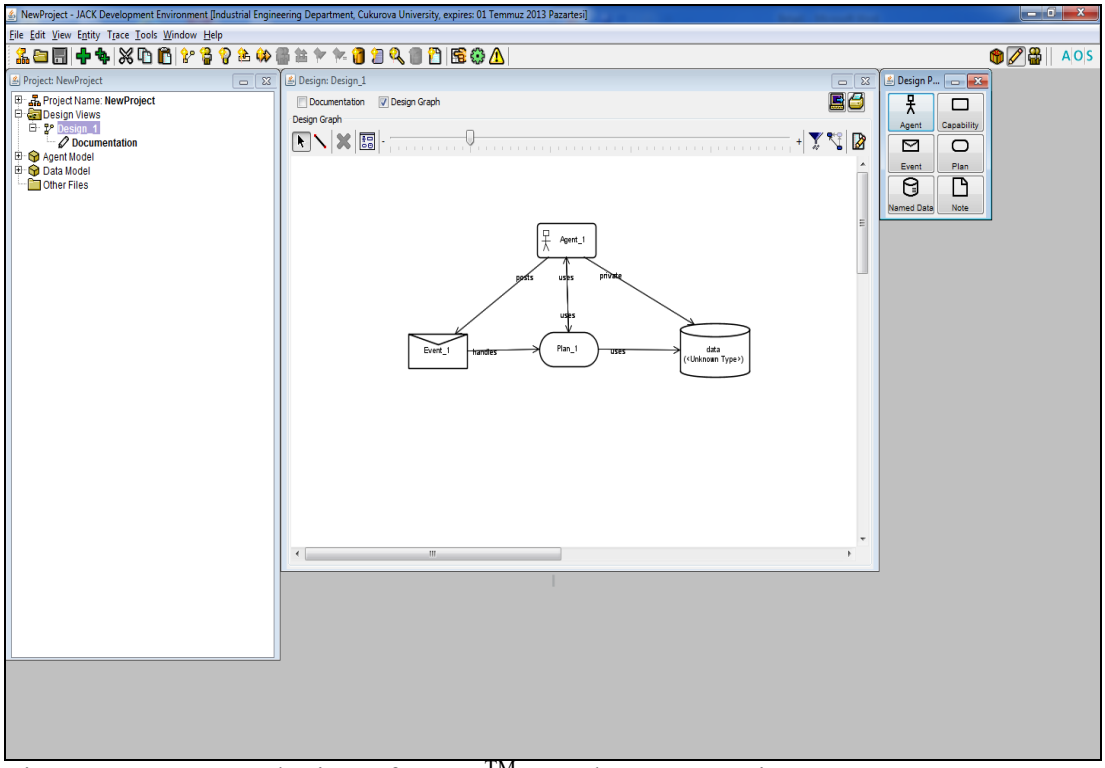


Figure 3.4. A General View of JACK™ Development Environment

JACK™ currently consists of three main extensions to to Java™. The first one is a set of syntactical additions to its host language. These additions, in turn, can be divided as follows (Howden et al, 2001; AOS, 2009):

- A small number of keywords for the identification of the main components of an agent (such as agent, plan and event);
- A set of statements for the declaration of attributes and other characteristics of the components (for instance, the information contained in beliefs or carried by events). All attributes are strongly typed;
- A set of statements for the definition of static relationships (for instance, which plans can be adopted to react to a certain event);
- A set of statements for the manipulation of an agent's state (for instance, additions of new goals or sub-goals to be achieved, changes of beliefs, interaction with other agents).

As it is described above the second extension to Java™ is a compiler that converts the syntactic additions into pure Java™ classes and statements. Afterwards, it can be loaded with and called by another Java™ code. In addition, the compiler also partially transforms the code of plans in order to obtain the correct semantics of the BDI architecture.

Finally, a set of classes (called the kernel) provides the required run-time supporting the generated code. This includes:

- Automatic management of concurrency among tasks being pursued in parallel (Intentions in the BDI terminology);
- Default behavior of the agent in reaction to events, failure of actions and tasks, and so on;
- Native lightweight, high performance communications infrastructure for multi-agent applications.

JACK™ extends the BDI paradigm to deal with inter-agent co-ordination, and includes new constructs and tools that support modern software engineering practices. JACK™'s key programming constructs are outlined in Table 3.1.

Table 3.1. Key JACK™ Constructs (AOS, 2009)

Programming Construct	Description
Event	Events are the central motivating factor in agents. Without events, the agent would be in a state of torpor, unmotivated to think or act. Events can be generated in response to external stimuli or as a result of internal computation. The internal processing of an agent generates events that trigger further computation.
Plan	Plans are procedures that define how to respond to events. When an event is generated, JACK™ computes the set of plans that are applicable to the event and selects the plan that will form its next intention. Plans have a body that defines the steps to be executed in response to the event. The agent can try alternative plans to achieve the same goal.
Beliefset	Beliefsets are used to represent the agent's declarative beliefs – what it knows about itself and the world.
Agent	Agents are autonomous computational entities with their own external identity and private internal state.
Team	Teams are used to encapsulate the co-ordinated aspects of (multiple) agent behavior.

Agent Oriented Software is pursuing an aggressive release schedule for JACK™, which is constantly extending the software suite with new features and tools. In order to support these developments, Agent Oriented Software is involved in multiple research projects working towards the advancement of JACK™ and agent systems in general (Howden et al., 2001).

4. PROPOSED SYSTEM

In the current study, the problem related to the published Flexible Manufacturing Systems in literature is solved through an agent based model with the proposed framework model. The proposed system takes operation flexibility of the machines into consideration. In this model, the orders reaching the system are defined as an agent and the schedule of the system is dynamically created with simple rules. This attribute for the proposed model increases the responsiveness of the flexible manufacturing systems. In addition, the proposed model also provides a responsive scheduling approach which is adaptive to the dynamic nature of the manufacturing systems.

4.1. Design of Proposed Multi-Agent System

Agent-based modeling and designing systems differ from conventional systems design. In this study, agent-based flexible production management system is designed by Prometheus^{TM} methodology. Expected functionalities are realized by the method. In this method, the manager gives orders to the system as well as adding and removing resources to the system dynamically. Moreover, the manager determines the duration of the order of operations and assignment of operations to the machines along with Automated Guided Vehicles which are called material handling system components. Agent-based systems are designed in three stages by means of Prometheus^{TM} methodology; system specifications, architecture design and detailed design.

4.1.1. System Specifications

In the first part of Prometheus^{TM} method, the system specification stage, negotiations among the types of agents, the goals of the system, agent roles within the system and scenarios of the system are determined. Therefore, system goals and sub-goals are defined as:

- Import and submit orders dynamically
    - Determine operations of order
    - Obtain order information
- Dynamic control of system resources
    - Add and remove resources to the system (Machine or AGV)
    - Determine the situation of system resources
    - Start the system
- Assign operations to machines
    - Process assigned jobs
    - Prepare and submit a bid for the processing of related operation of the job
        - Create machine schedules
- Assign operations to AGVs
    - Transport assigned jobs
    - Prepare and submit a bid for the transportation of related operation of the job
        - Create AGV schedules
- Negotiation based on job assignment
    - Negotiate between the AGVs in the system
    - Negotiate between the machines in the system

System scenarios defined as:

- Negotiate between the AGVs scenario
- Negotiate between the Machines scenario
- Create AGV schedules scenario
- Import and submit orders dynamically scenario
- Dynamic control of system resources scenario
- Assign operations to Machines scenario
- Assign operations to AGVs scenario

- Negotiations based on job assignment scenario

The functionality of Negotiation Management, AGV Management, Machine Management and System Management are defined as functionalities within system specification panel in Figure 4.1. At this stage, the system startup information is designed as percepts, such as new orders demand and change of resource capacity. The system scenarios determine whether or not the roles are active along with percepts which are connected to System Management. Descriptions of the goals, functionalities and roles are seen in Figure 4.1.

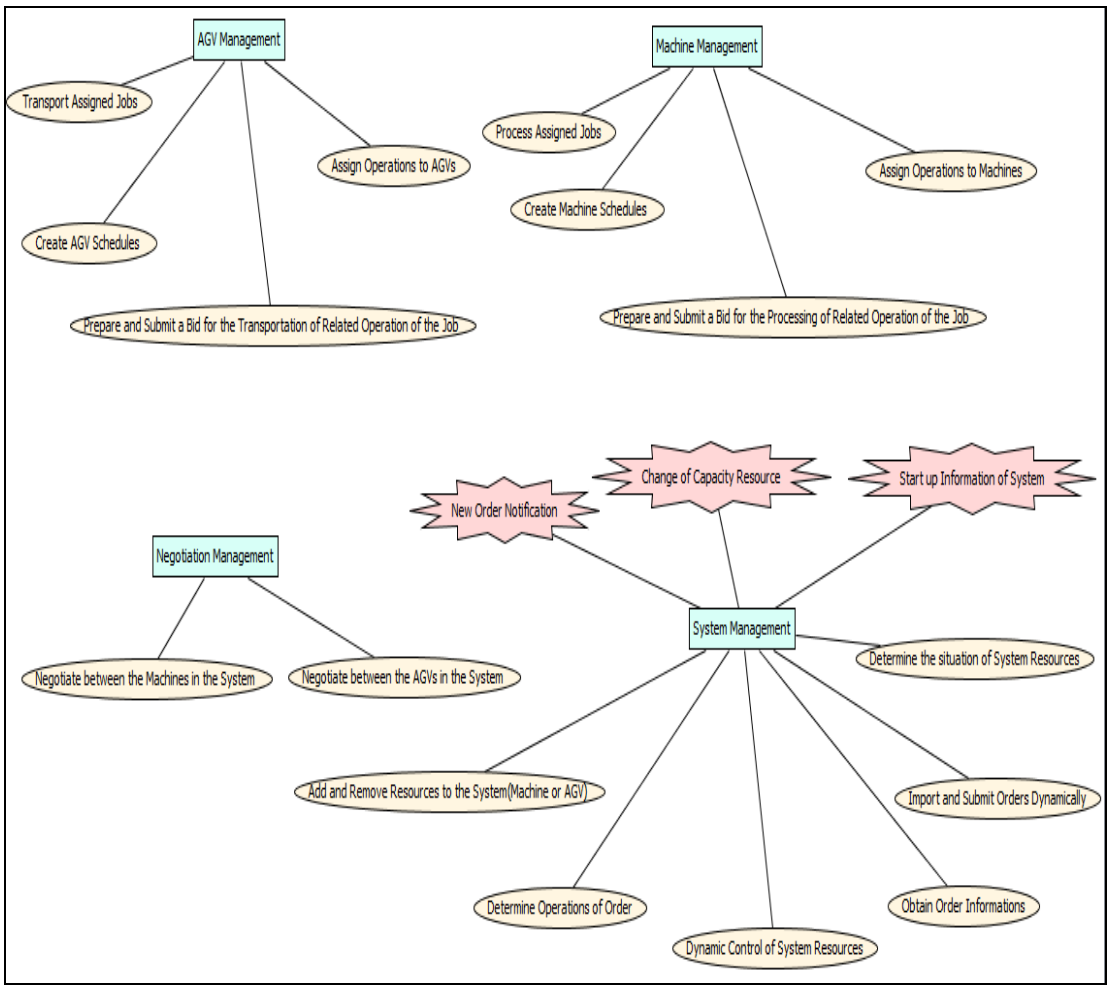


Figure 4.1. PDT System Role Definitions

4.1.2. Architecture Design

This stage includes the identification of agent types according to system specification phase in which the roles of agents in the system are determined. After determining the agent types, negotiations among the agents are defined in the protocol packets. Agents and their relationships are shown in Figure 4.2.

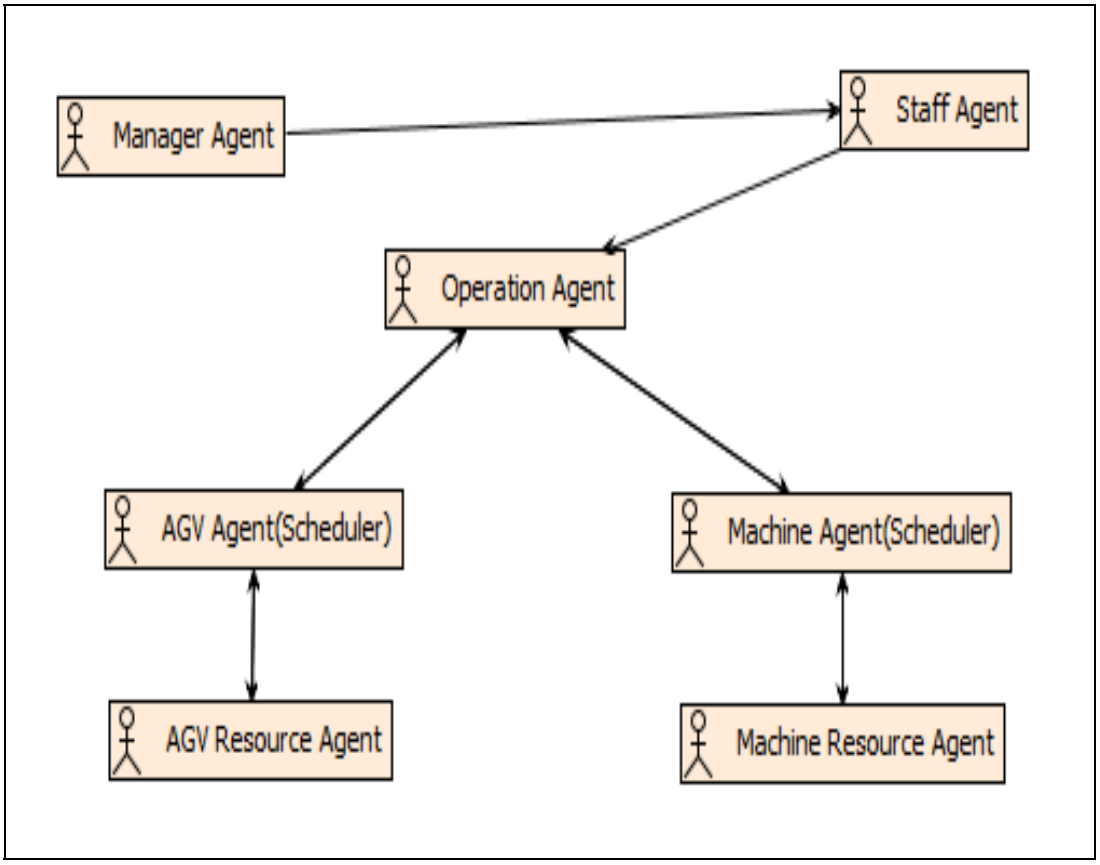


Figure 4.2. PDT Agent Types

In order to solve events, agents can get in touch with each others and act proactively in syncronization. This properity is supplied by protocols. In Figure 4.3, agents and protocols in which messages are defined among agents are seen.

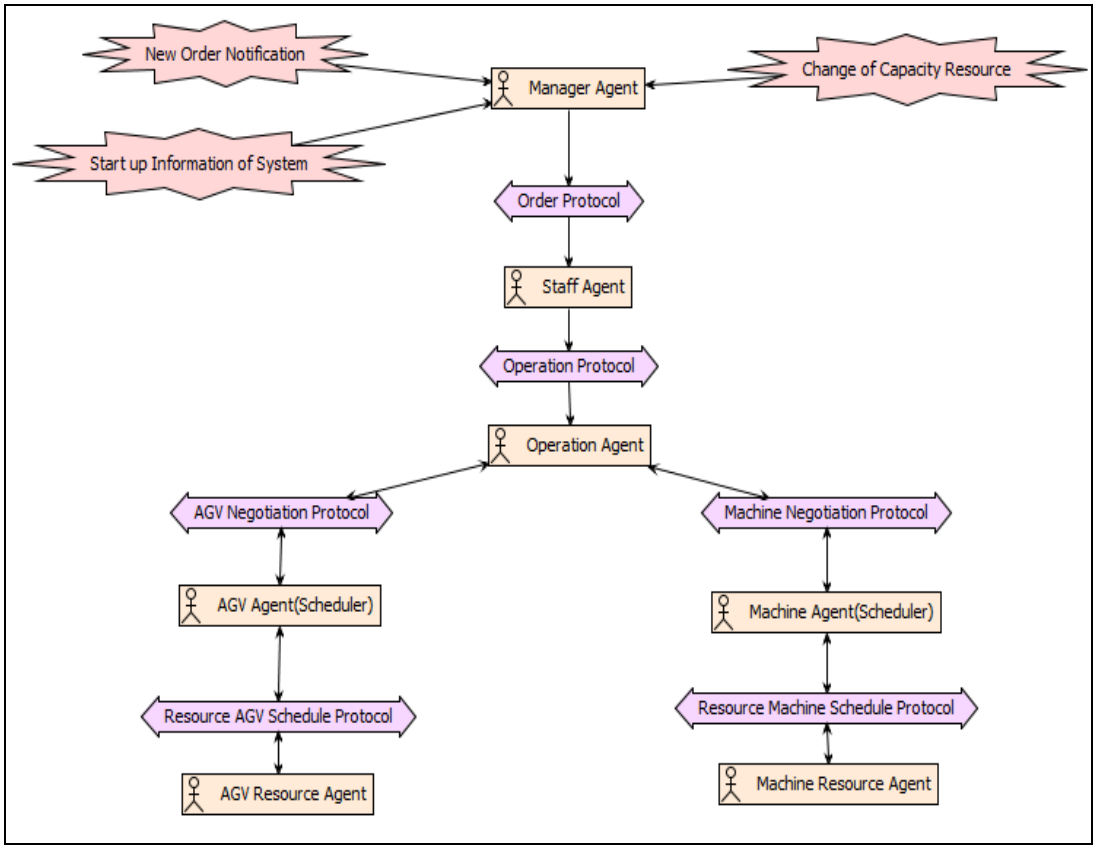


Figure 4.3. PDT Agent Overview

In Figure 4.3, protocols are constructed as Order Protocol, Operation Protocol, AGV Negotiation Protocol, Resource AGV Schedule Protocol, Machine Negotiation Protocol and Resource Machine Schedule Protocol.

This negotation protocols are coded in Prometheus™ software. Figures of protocols are demonstrated by means of texts and diagrams. The negotiation protocol is given between Operation Agent and Machine Agent in Figure 4.4. In this protocol, *Create_Request_Process, Start_Process_Negotiation, Report_Winner_Machine* messages are sent to Machine Agent(Scheduler) by Operation Agent., *Send_Bidding_Process, Send_Completed_Process* messages are sent to Operation Agent by Machine Agent(Scheduler).

```
start Machine Negotation Protocol
agent A Operation Agent
agent B Machine Agent(Scheduler)
message A B Create_Request_Process
box alt
message A B Start_Process_Negotiation
next
message B A Send _Bidding_Process
next
message A B Report_Winner_Machine
next
message B A Send_Completed_Process
end alt
finish
```

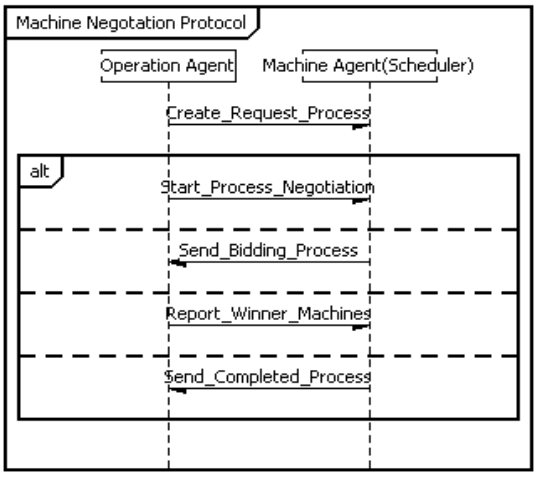


Figure 4.4. Machine-Operation Agent Negotiation Protocol

The negotiation protocol betwwen Machine and Machine Resource Agens is also given in Figure 4.5. According to this protocol, *Machine_Send_Schedule* message is sent to Machine Resource Agent by Machine Agent(Scheduler). The answer is sent by means of *Machine_Idle_Information* message to Machine Agent(Scheduler). The codes constructing the diagram for this protocol is also given inside of the figure. Other protocols used by the system is also built in the same manner.

```
start Resource Machine Schedule Protocol
agent A Machine Agent(Scheduler)
agent B Machine Resource Agent
message A B Machine_Send_Schedule
box alt
message B A Machine_Idle_Information
end alt
finish
```

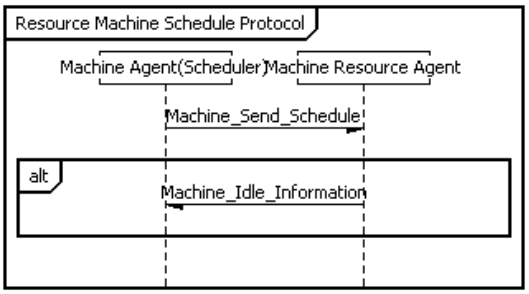


Figure 4.5. Machine- Machine Resource Agent Negotiation Protocol

4.1.3. Detailed Design

Detailed design is done for each of the agents one by one. Types of agents at this stage take the message from the event of their environment or other agents which operate on their plans, and thus they act according to the record in their database.

Manager Agent starts up the system dynamically. Manager Agent uses its beliefsets, plans and message events so that it will accomplish this task. Afterwards, the other agent types are created in the system. Manager Agent architecture is given in Figure 4.6 in the form of Prometheus™ design view.

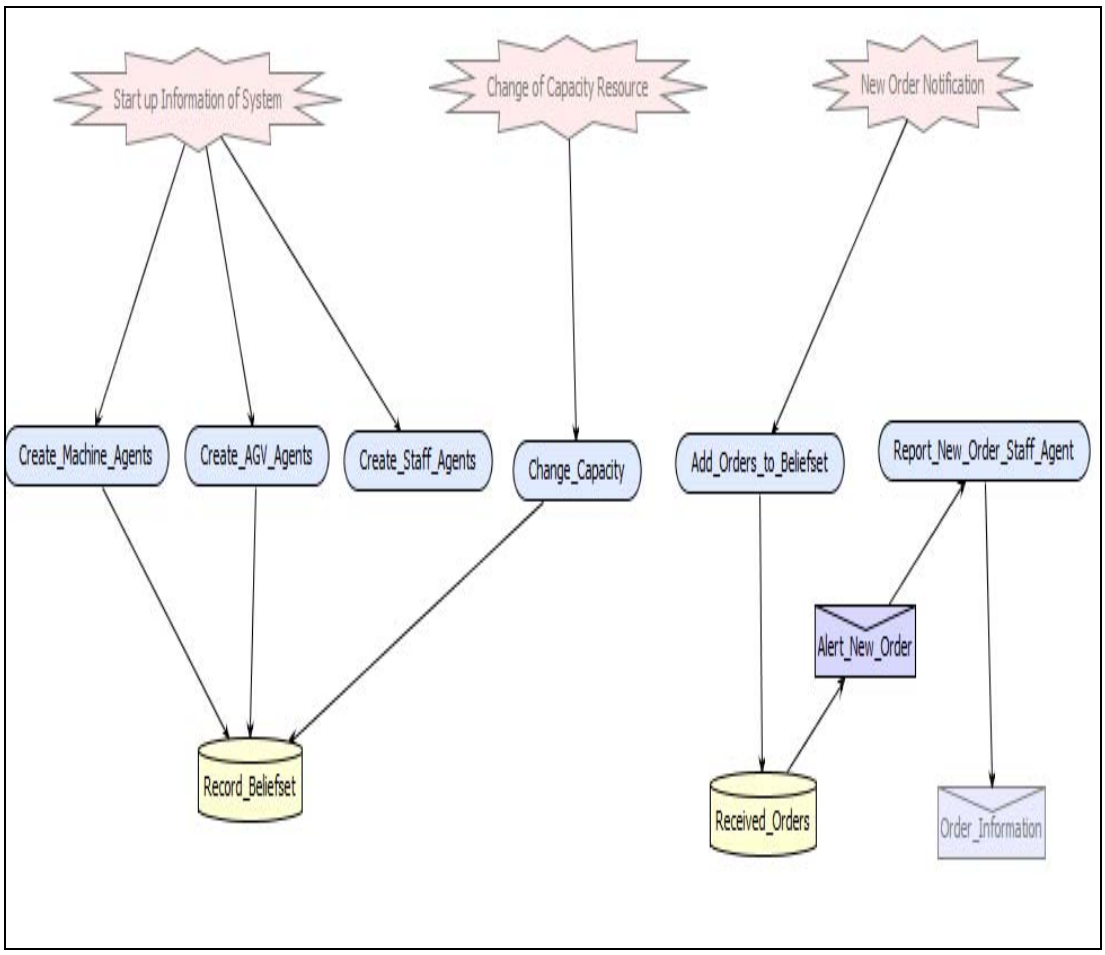


Figure 4.6. Detailed Design of Manager Agent

Manager Agent firstly creates Staff Agent, AGV Agents (Scheduler and Resource) and Machine Agents (Scheduler and Resource) according to *start up Information of System* percept which comes from external environment. Manager Agent performs its local plans which are *Create_Machine_Agents, Create_AGV_Agents, and Create_Staff_Agents*. During the agent creation process, the agent types are registered to *Record_Beliefset* (see Figure 4.6). Resource changes within the system are informed by means of Change of Capacity Resource Percept

and then necessary modifications are made within the beliefset. Whenever a new order is created, it is added to order *Received_Orders* beliefset by means of *Add_Orders_to_Beliefset* plan. *Alert_New_Order* message including order related information is posted automatically and handled by *Report_New_Order_Staff_Agent* plan. *Order_Information* message is sent to Staff Agent to trigger for its cycle.

When a new order is sent to Staff Agent by Manager Agent, Staff Agent creates the relational operation agents. In the Figure 4.7, The Staff Agent receives the *Order_Information* message from Manager Agent and creates the respective Operation Agents. Staff Agent uses its *Create_Operation_Agents* plan. Then, it sends Operation*_Information* message to the Operation Agent.

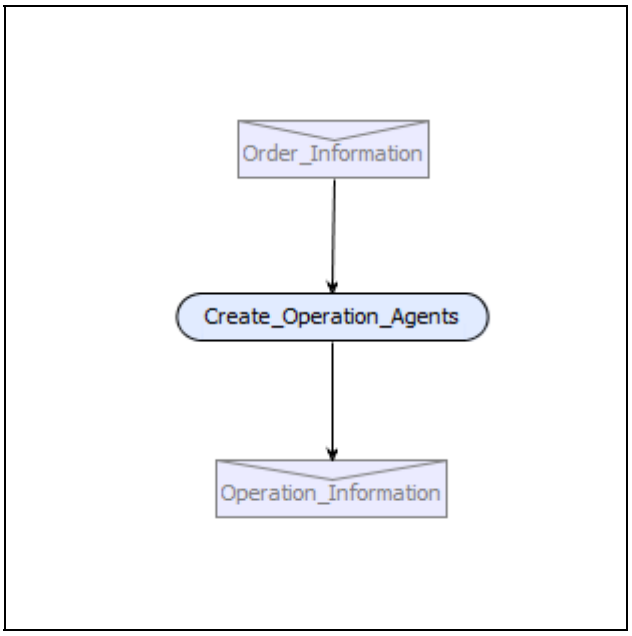


Figure 4.7. Detailed Design of Staff Agent

Every Operation Agent has two duties within the agent system which are: transportation and processing. Operation Agent architecture is given in Figure 4.8 which shows the detailed design of the type of operation agent, the received messages and sent messages. Beside this, it includes beliefsets for agents.

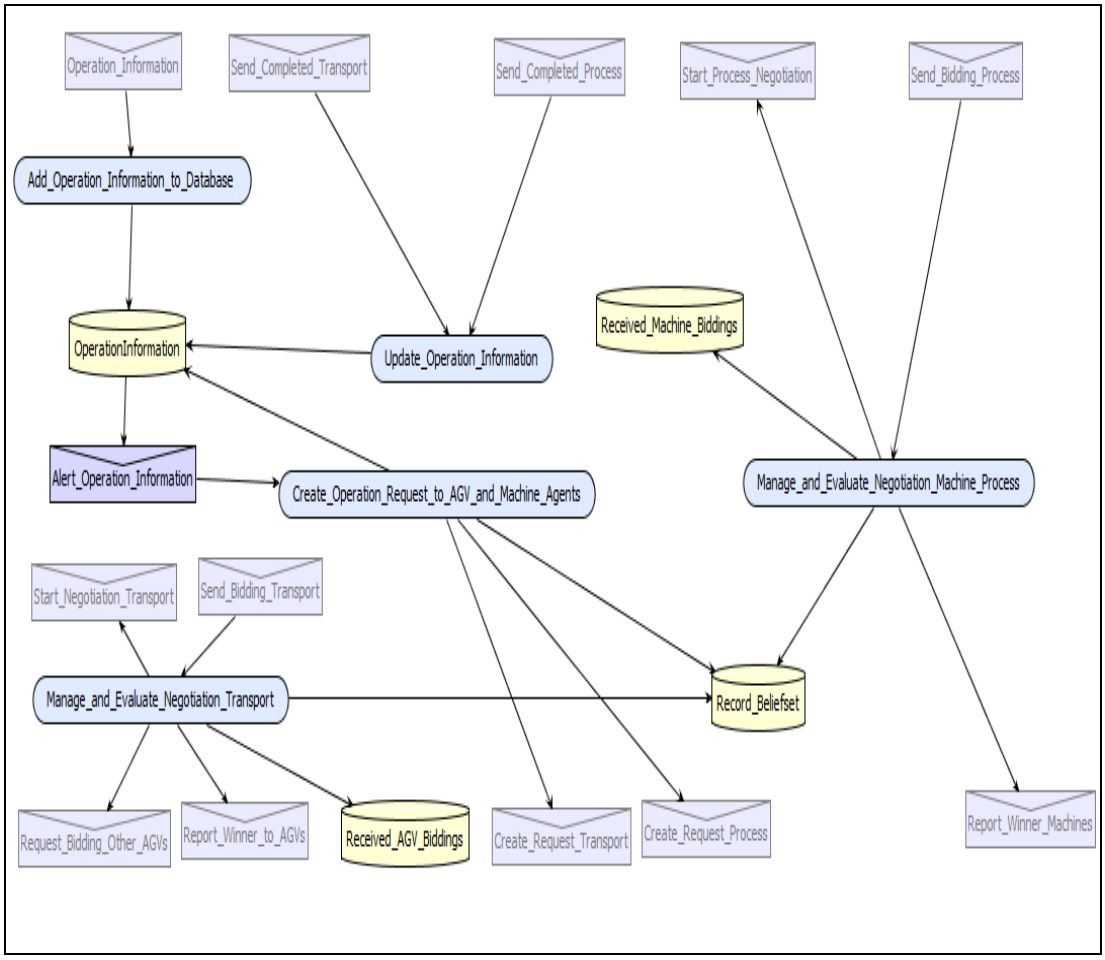


Figure 4.8. Detailed Design of Operation Agent

After the operation information is sent by Staff Agent, Operation Agent firstly records operation information to OperationInformation beliefset by using *Add_Operation_Information_to_Database* plan. *Alert_Operation_Information* event including operation related information is posted automatically and handled by *Create_Operation_Request_to_AGV_and_Machine_Agents* plan. If it is the first operation, the plan starts with sending a transportation request to all AGV Agents and registers them to *Record_Beliefset*; otherwise, it may wait for the completion of previous operations. Then, the messages, *Create_Request_Transport* and *Create_Request_Process* are sent to AGV and Machine Agents, respectively.

When every operation is created by Staff Agent, this architecture including plans, events, and beliefsets are also created and ready to use. Firstly, *Start_Process_Negotiation* message is sent to the Machine Agents available in the

system and they submit an offer to operate the job by means of *Send_Bidding_Process* message. These events are sent and handled by *Manage_and_Evaluate_Machine_Process* plan of Operation Agent and negotiation protocol is started in this plan (See Appendix A). Machine Agents in the system check their situation and beliefsets. The result of the bid is sent to winner Machine Agent as the best proposal via *Report_Winner_Machines* message by Operation Agent. In the same manner, *Start_Negotiation_Transport* message is sent to the AGV Agents for asking to submit an offer to transport by means of *Send_Bidding_Transport* message. Finally, when transport and process are completed, Operation Agent receives a *Send_Completed_Process* message from Machine Agent and a *Send_Completed_Transport* message from AGV Agent. This message is handled by *Update_Operation_Information* plan of Operation Agent.

In the AGV Agent architecture, there are some messages, which activate the AGV Agents (see Figure 4.9). When an operation is started on a machine, a transportation request by the consecutive operation agent is made over *Create_Request_Transport* event. This information is recorded by *Add_Transport_Request_to_the_Blackboard* plan of AGV Agent and used by *Check_AGV_Blackboard_and_Search_Suitable_Job* plan for calculating the most appropriate task. When the plan is triggered by *AGV_Idle_Information* message, it is checked whether any task is rewarded according to the AGV Resource Agent's state. If there is a rewarded transportation task, this job information is sent to AGV Resource Agent with *AGV_Send_Schedule* event and the plan finishes (Şahin, 2010; Erol et al., 2012).

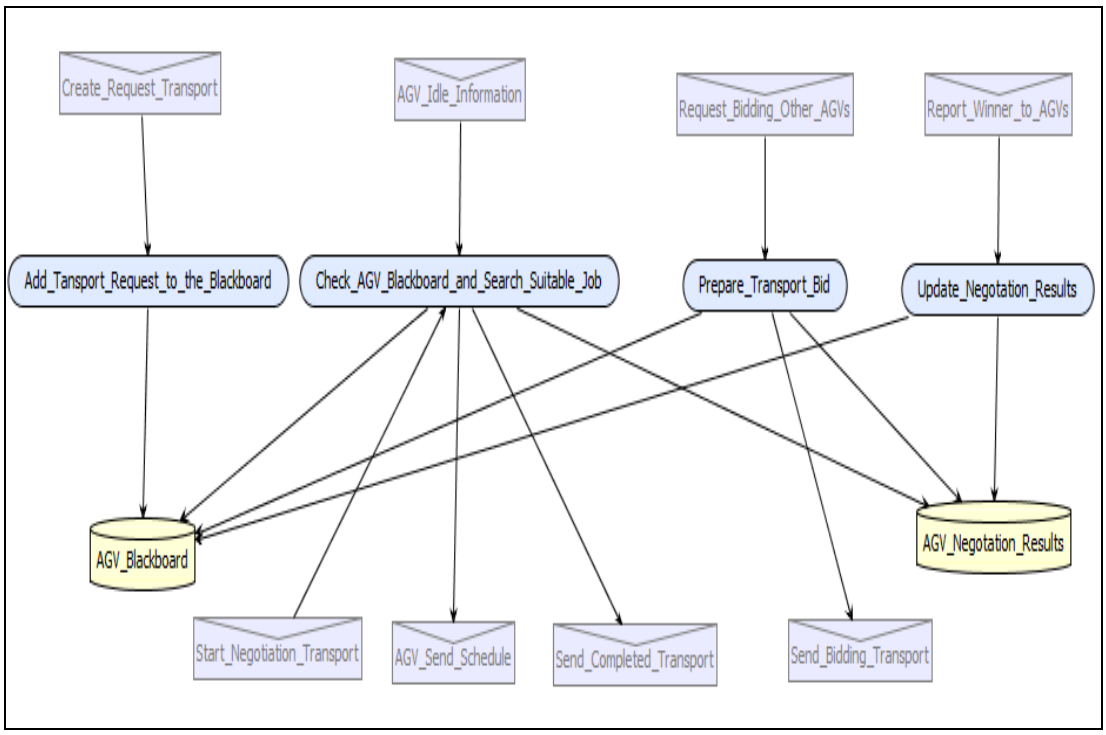


Figure 4.9. Detailed Design of AGV Agent(Scheduler)

On the other hand, if there is not a rewarded job for AGV, AGV search on the blackboard to select a job which has the lowest cost. This procedure and formulas to select the best job is given in the study of Erol et al. 2012. Earliest loading time can be determined; if current time is later than the earliest pick up time of the tasks, earliest loading time of *n* tasks in the *AGV_Blackboard* is calculated using equation 4.1, Where ELT_i denotes the earliest loading time of task *i*, *CL* is current location of AGV Resource Agent, PCP_i is pick up point of task *i*, *t* is current time, $\Delta t\ (.,.)$ is the time difference between two locations and EPT_i is earliest pickup time of task *i* (Erol et al., 2012).

$$ELT_i = \begin{cases} t + \Delta t(CL, PCP_i), & t > EPT_i,\ i = 1 \ldots n \\ t + max\{\Delta t(CL, PCP_i), (EPT_i - t)\}, & t \le EPT_i, i = 1 \ldots n \end{cases} \quad (4.1)$$

Then, the selected task from AGVBlackboard (ELT_s) is found by using equation 4.2 (Erol et al., 2012).

$$ELT_s = \underset{i=1\ldots n}{Min}(ELT_i) \tag{4.2}$$

The AGV Agent proposes to respective operations by adding ELT_s to relational loaded trip time. Therefore, AGV Agent can make the reasoning of selection the most appropriate task to transport and to minimize its idle time. This statement is given in equation 4.3, where *s* represents the selected task and *PR* represents the proposal (Erol et al., 2012).

$$PR = ELT_s + \Delta t(PCP_s, DP_s) \tag{4.3}$$

After the proposal is calculated in this plan, it is sent to relational Operation Agent by means of *Send_Bidding_Transport* event and the bidding is started by the Operation Agent as described earlier and the plan is suspended until the bidding's results is received. After the start of bidding, Operation Agent asks the other AGV Agents whether they are idle or not to submit their proposals. The agents participating the bidding check their blackboards and submit a proposal using *Prepare_Transport_Bid* plan (Erol et al., 2012).

The plan firstly checks whether there is a task already rewarded. If there is not a rewarded task, it prepares an offer; otherwise, it does not offer any proposal. When preparing a proposal, AGV Agent makes a search on its blackboard to find the task which has minimum *ELT* by considering earliest pickup time, earliest free time of AGV Agent and the last location of AGV. If the earliest free time of AGV Agent, in when AGV Agent becomes idle, is later than the earliest pick up time of the tasks, earliest loading time of n tasks in the *AGV_Blackboard* is calculated using the equations 4.4 and 4.5, where *EFT* denotes earliest free time of AGV Agent and *NL* denotes the next location of AGV Agent (Erol et al., 2012).

$$ELT_i = \begin{cases} EFT + \Delta t(NL, PCP_i), & EFT > EPT \quad ,i = 1\ldots n \\ EFT + max\{\Delta t(NL, PCP_i), (EPT_i - EFT)\}, & EFT \le EPT, i = 1\ldots n \end{cases} \tag{4.4}$$

$$ELT_s = \underset{i=1\ldots n}{Min}(ELT_i) \tag{4.5}$$

The AGV agent prepare proposal by means of equation 4.6 if the selected task of AGVBlackboard is the same as the bidding task by adding relational loaded trip time to ELT_s (Erol et al., 2012).

$$PR = ELT_s + \Delta t(PCP_s, DP_s) \tag{4.6}$$

When the negotiation is over, the result is declared by Operation Agent using *Report_Winner_to_AGVs* message. This event is handled by *Update_Negotiation_Results* plan and result is recorded in *AGV_Negotiation_Results* beliefset. Whenever AGV Resource Agent becomes idle, it informs AGV Agent by sending *AGV_Idle_Information* message. After that, the AGV Agent sends a message to inform the completion of transportation to Operation Agent by sending *Send_Completed_Transport* message (Erol et al., 2012).

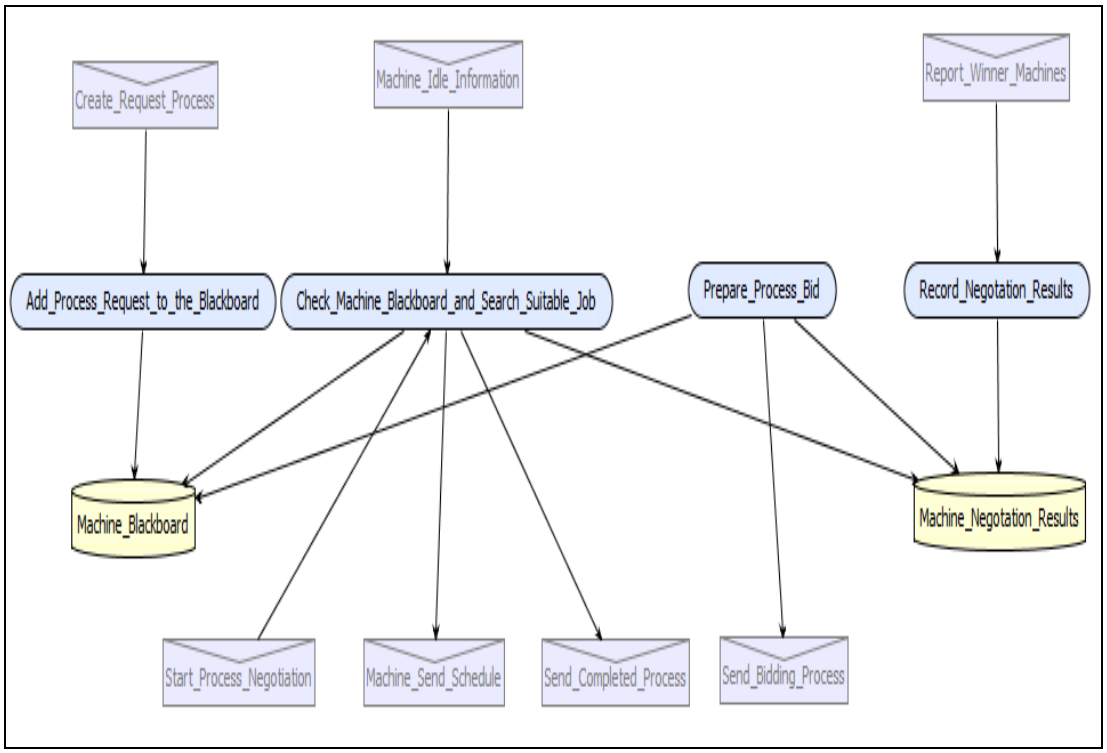


Figure 4.10. Detailed Design of Machine Agent(Scheduler)

In the Machine Agent architecture, there are some messages, which activate the Machine Agents (see Figure 4.10). When an operation is started on a machine, a transportation request by the consecutive operation agent is made over *Create_Request_Process* event. This information is recorded by *Add_Process_Request_to_the_Blackboard* plan of Machine Agent and used by *Check_Machine_Blackboard_and_Search_Suitable_Job* plan for calculating the most appropriate task (See Appendix B). When the plan is triggered by *Machine_Idle_Information* message, it is checked whether any task is rewarded according to the Machine Resource Agent's state. If there is a task rewarded in the blackboard, this job information is sent to Machine Resource Agent with *Machine_Send_Schedule* event and the plan finishes.

After receiving the job's knowledge via messages, by considering their work load and distance from the work, machine scheduling agents make a bid by using equation 4.7. Two costs are calculated in *Machine_Blackboard* and these costs are sent to Operation Agent.

$$\begin{aligned} TIY_{ij} &= \sum_{i=1}^{n} p_{ij} \quad , \qquad j = 1 \ldots\ldots k \\ AS_{ij} &= p_{ij} + \Delta t(ML_j, PML_i), \quad j = 1 \ldots\ldots k \end{aligned} \tag{4.7}$$

TIY_{ij} is the total task load of machine j, if task i is processed by machine j. AS_{ij} is the total flow time of task i, if task i is processed by machine j (Except the waiting time). p_{ij} is the processing time of task i by machine j. n is the total task number. ML_j is the location at which Machine Agent is. t is the current time, PML_i is the location of the former machines by which task i is processed, k is the total machines number, $\Delta t(.,.)$ is the expression of the time difference between two machines.

After receiving the proposed TIY_{ij} and AS_{ij} values, Operation Agent standardizes these in the rage 0-1 by using equations 4.8 and 4.9. Then, it calculates the total cost for each machine by using equation 4.10.

$$STDTIY_j = \frac{\underset{j=1...k}{Max}(TIY_j) - TIY_j}{\underset{j=1...k}{Max}(TIY_j) - \underset{j=1...k}{Min}(TIY_j)}, \qquad j = 1.....k \tag{4.8}$$

$$STDPML_j = \frac{\underset{j=1...k}{Max}(AS_j) - AS_j}{\underset{j=1...k}{Max}(AS_j) - \underset{j=1...k}{Min}(AS_j)}, \qquad j = 1.....k \tag{4.9}$$

$$TM_j = STDPML_j + STDTIY_j, \quad j = 1.....k \tag{4.10}$$

$STDTIY_j$ is the standardized total task load of machine j, $STDPML_j$ is the standardized total flow time for machine j, TM_j is the total cost calculated for each machines and k is the total number of Machine Agents.

Then, as mentioned in the conceptual model, Operation Agent rewards the Machine Agent which has minimum TM_j through equation 4.11 and sends the information on relevant jobs to the Machine Agent. In this way, Machine Agent succeeds in the minimization of its own jobs.

$$SM = \underset{j=1...k}{Min}(TM_j), \quad j = 1.....k \tag{4.11}$$

When the negotiation is over, the result is declared by operation agent using *Report_Winner_to_Machines* message. This event is handled by *Record_Negotiation_Results* plan and the result is recorded in *Machine_Negotiation_Results* beliefset. Whenever Machine Resource Agent becomes idle, it informs Machine Agent by sending *Machine_Idle_Information* message. After that, Machine Agent sends a message to inform the completion of related process to Operation Agent by sending *Send_Comleted_Process* message.

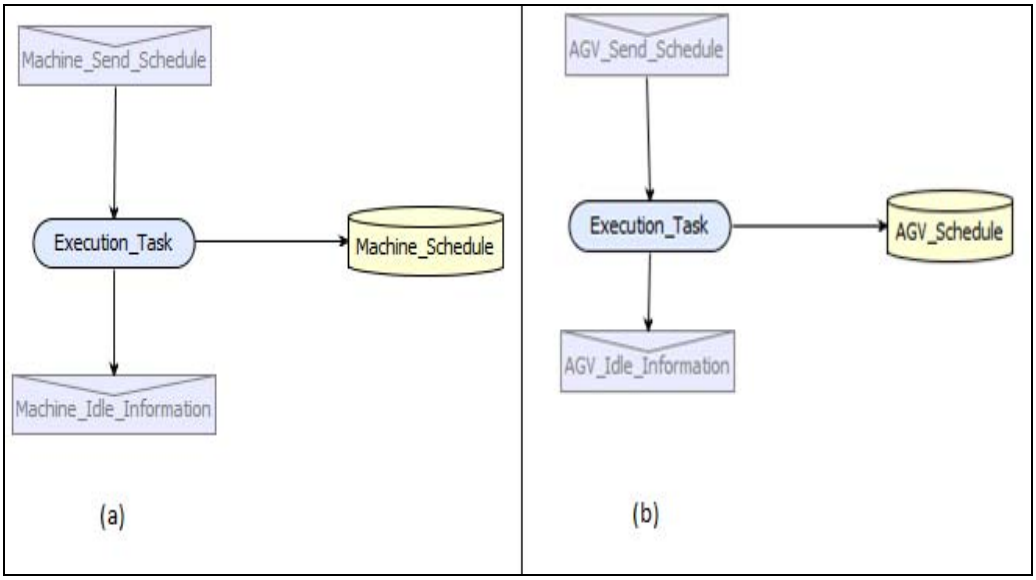


Figure 4.11. Detailed Design of Machine and AGV Resource Agents

AGV and Machine Resource Agents represent the physical resources of AGVs and machines. These agents have only one plan and one beliefset. After receiving the new task from the AGV Agent or Machine Agents via *AGV_Send_Schedule and Machine_Send_Schedule* messages, they start to carry the order to the machine in progress. After the completion of task, they record the completed task in their beliefsets and ask AGV or Machine Agents to send a new task by means of *AGV_Idle_Information* and *Machine_Idle_Information* messages. This process is executed via Execution_Task plan. The architectures of Machine and AGV resource agents are depicted in Figure 4.11 (a) and (b) respectively.

After, three stages of the Prometheus™ Design tool are accomplished, Java™ codes can be generated in JACK™. Obtained JACK™ results are used to benchmark the proposed system performance with other results in literature.

4.2. Implementation of the Proposed Multi-Agent System in JACK™

In this study, the system is designed by Prometheus™ and implemented through JACK™ agent development environment. The main objectives of JACK™ components are the same as Prometheus™ design tool. In the proposed system, after the creation of Manager Agent, it starts sending jobs (In Figure 4.12).

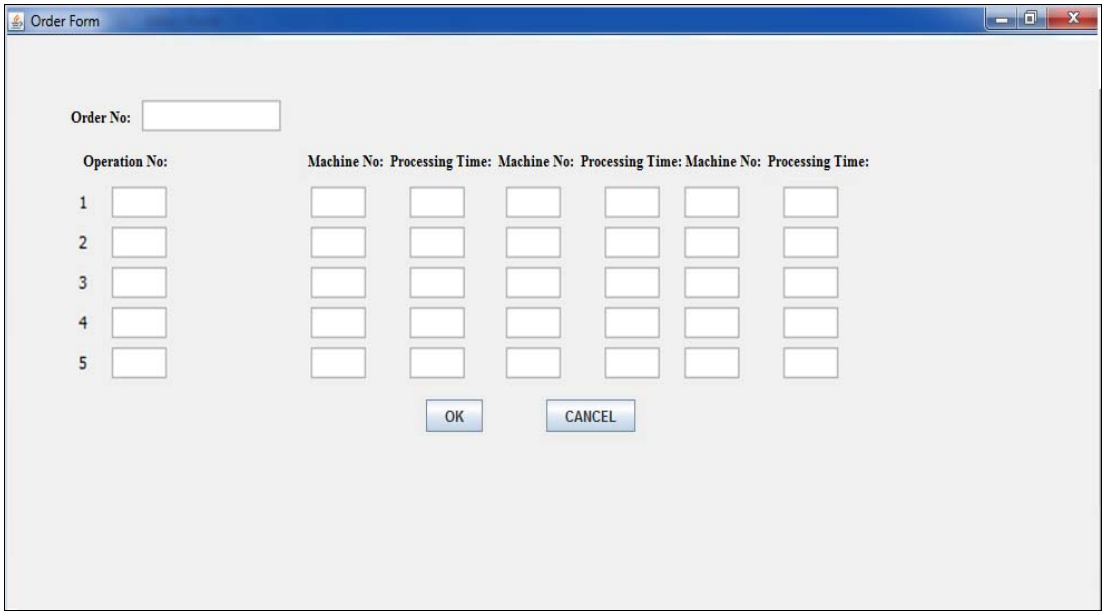


Figure 4.12. Create Order Form in the Multi-Agent System

Manager Agent creates a list of operations for the newly created job syncronically. Then, an available operation is chosen from the list, as well as the identification of the business route. When a job is created, order information on the job is sent to the Staff Agent. Received information about the new order is sent to Operation Agents by Staff Agent. Operation Agents cooperate with the required equipment which provides the transportation, and then communicate with AGV and Machine Agents which work for the completion of the work during the implementation of the operation on the list.

First of all, Manager Agent creates Staff Agent. In the proposed system, Staff Agent's plan uses the same way of the Erol's (2012) study. Flow chart of the plan in the form of JACK^{TM} plan editor is demonstrated in Figure 4.13 (Erol et al., 2012).

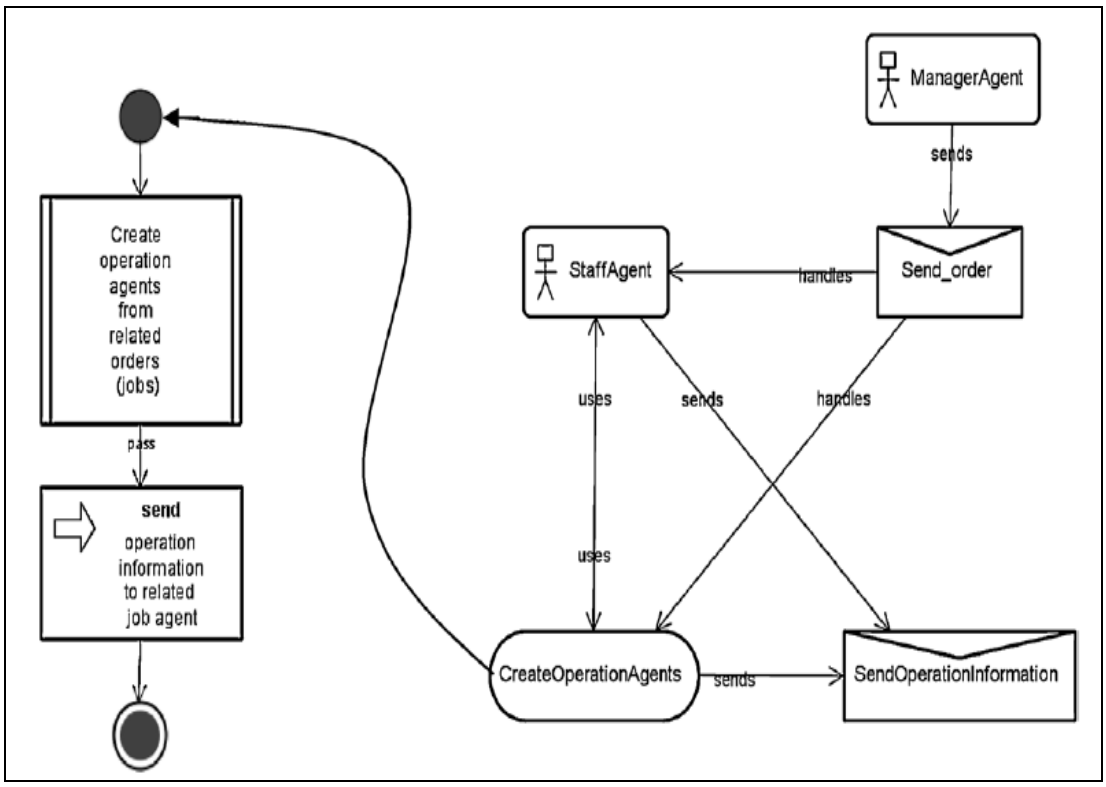


Figure 4.13. Staff Agent Architecture and Flow Chart of the CreateOperationAgents Plan of Staff Agent (Erol et al., 2012)

When operation information comes from Staff Agent, Operation Agent makes a request for bids. Firstly, Operation Agent sends operation information to Machine Agents. Machine Agents submit a proposal to Operation Agent. Operation Agent selects appropriate machines. Machine Agents reasoning flow is based on contract net protocol and it is seen in the form of JACK^{TM} plan editor in Figure 4.14. After selecting machines, the negotiation is started between AGV Agents by Operation Agent for transportation task as explained in previous section. AGV Agents submit an offer to transport the operation oforder between any two locations. AGV Agents send their proposals. Operation Agent selects the best proposals and declares the winners to AGV Agents.

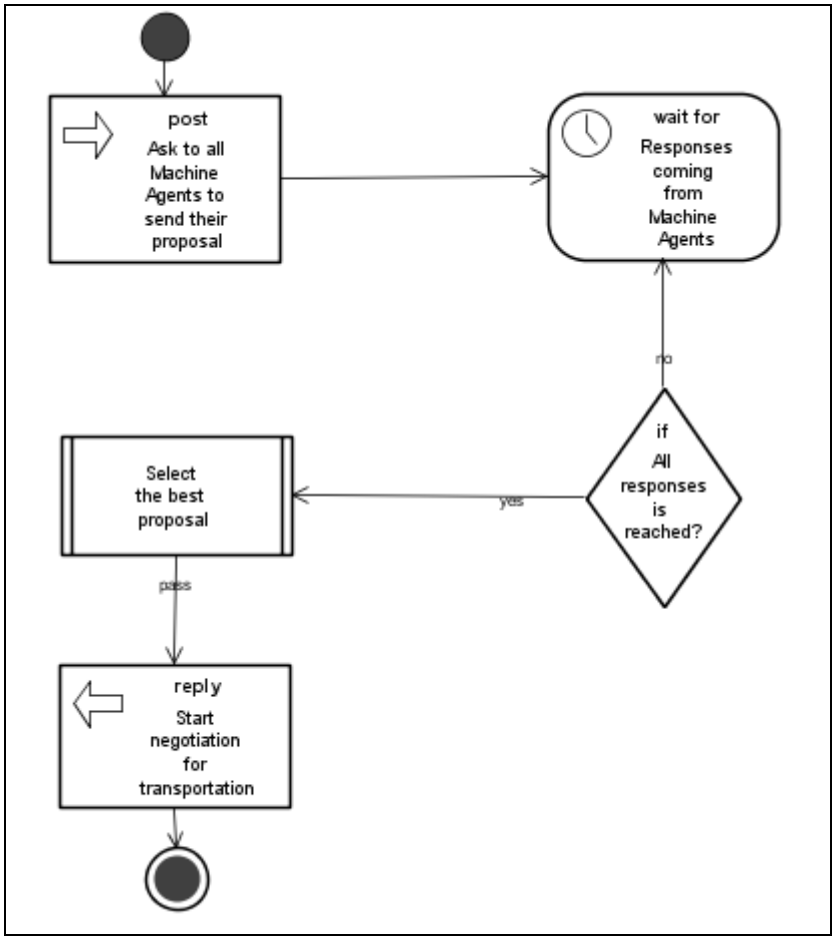


Figure 4.14. Machine Negotiation Plan of Operation Agent

AGV Agent and Machine Agent prepare their plan by checking their tasks. If there is any task rewarded, the task is sent to Resource Agents (AGV and Machine) for transportation or processing of relative operation. If there is not, AGV and Machine Agent check their Blackboard to find a suitable task. After reasoning completed, the proposal is sent to Operation Agent for selected task. When the bidding results are announced, agents execute their tasks.

5. RESEARCH FINDINGS

5.1. Test Study

In this section, a test study is used to analyse the performance of the proposed system. A test problem is chosen which was presented in the studies of Bilge and Ulusoy (1995) and Kumar et al. (2011). This problem was firstly defined by the Bilge and Ulusoy (1995) as offline scheduling problem and solved regardless of the machine flexibility. To benchmark the proposed system performance, the problem, orginally defined by Bilge and Ulusoy (1995) and extended by Kumar et al. (2011), is chosen. Kumar et al. (2011) used machine flexibility in their study and solved the problem by Differential Evulation heuristic method (FDE). Data sets in these studies are used to make a performance comparision among the models. As a performance criterion, the total production time (makespan) is selected. Offline optimization results provide an upper limit to the performance of the proposed multi-agent system.

Figure 5.1 represents layout configurations used in this study. The distances from load/unload station to machines and distances between a pair of machines are given in time units for all the four different layouts. When generating the problems, an important characteristic is taken into account which is the relative magnitude of the travel times and the processing times. These characteristics are represented as the ratio ($\bar{t}/\bar{p}$), where $\bar{t}$ is the average of all the nonzero entries in the travel time matrix and $\bar{p}$ is the average of all the processing times. The problems are catogorised into two groups; one with relatively low (t/p) ratio and the other with relatively high (t/p) ratio (Bilge and Ulusoy, 1995; Kumar et al., 2011).

Table 5.1 represents the travel time matrix for this particular problem. Data for the job sets used in the sample problems are given in Table 5.2, Table 5.3 and Table 5.4.

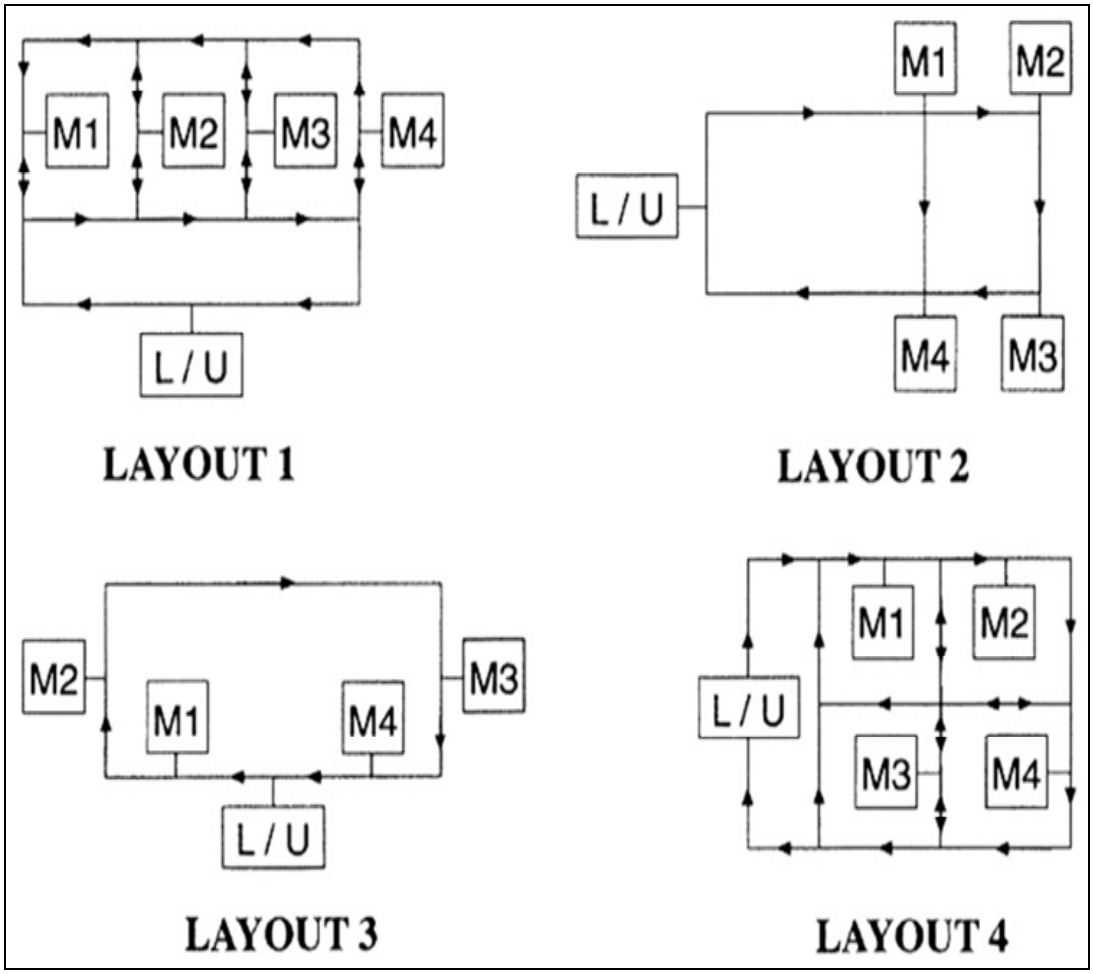


Figure 5.1. The Layout Configurations Used in the Example Problem (Bilge and Ulusoy, 1995; Kumar et al., 2011)

Table 5.1. Travel Time Matrix for the Example Problem (Bilge and Ulusoy, 1995; Kumar et al., 2011)

	L/U	M1	M2	M3	M4
Layout 1					
L/U	0	6	8	10	12
M1	12	0	6	8	10
M2	10	6	0	6	8
M3	8	8	6	0	6
M4	6	10	8	6	0
Layout 2					
L/U	0	4	6	8	6
M1	6	0	2	4	2
M2	8	12	0	2	4
M3	6	10	12	0	2
M4	4	8	10	12	0
Layout 3					
L/U	0	2	4	10	12
M1	12	0	2	8	10
M2	10	12	0	6	8
M3	4	6	8	0	2
M4	2	4	6	12	0
Layout 4					
L/U	0	4	8	10	14
M1	18	0	4	6	10
M2	20	14	0	8	6
M3	12	8	6	0	6
M4	14	14	12	6	0

Table 5.2. Data for the Job Sets Used in Example Problems (Kumar et al., 2011) (Part1)

Job set 1	**Job set 2**
Job 1: M1(8); M2(16); M4(12)	Job 1: M1(10); M4(18)
ALT1: M2(9); M3(14); M1(13)	ALT1: M3(12); M2(16)
ALT2 :M3(9); M4(17); M2(10)	ALT2: M2(11); M3(17)
Job 2: M1(20); M3(10); M2(18)	Job 2: M2(10); M4(18)
ALT1: M3(18); M1(13); M4(17)	ALT1: M4(8); M1(20)
ALT2: M2(21) ;M4(8) ;M3(19)	ALT2: M3(12); M2(16)
Job 3: M3(12); M4(8); M1(15)	Job 3: M1(10); M3(20)
ALT1: M4(11); M2(10); M3(14)	ALT1: M3(12); M2(18)
ALT2: M1(11); M3(7); M2(17)	ALT2: M2(8); M4(22)
Job 4: M4(14); M2(18)	Job 4: M2(10); M3(15); M4(12)
ALT1: M1(16); M3(16)	ALT1: M4(11); M1(17); M2(9)
ALT2: M3(16); M1(16)	ALT2: M1(13); M4(13); M3(11)
Job 5: M3(10); M1(15)	Job 5: M1(10); M2(15); M4(12)
ALT1: M2(9); M4(16)	ALT1: M4(12); M1(14); M3(11)
ALT2: M4(11); M3(14)	ALT2: M2(8); M3(14); M1(15)
	Job 6: M1(10); M2(15); M3(12)
	ALT1: M3(11); M4(13); M2(13)
	ALT2: M2(12); M3(14); M4(11)
Job set 3	**Job set 4**
Job 1: M1(16); M3(15)	Job 1: M4(11); M1(10); M2(7)
ALT1: M3(14); M1(17)	ALT1: M3(10); M2(9); M4(9)
ALT2: M2(15); M4(16)	ALT2: M2(13); M4(9); M3(6)
Job 2: M2(18); M4(15)	Job 2: M3(12); M2(10); M4(8)
ALT1: M1(17); M3(16)	ALT1: M1(10); M4(11); M3(9)
ALT2: M3(16); M2(17)	ALT2: M4(11); M3(9); M2(10)
Job 3: M1(20); M2(10)	Job 3: M2(7); M3(10); M1(9); M3(8)
ALT1: M3(19); M4(11)	ALT1: M4(10); M2(9); M3(8); M1(7)
ALT2: M4(19); M1(11)	ALT2: M3(9); M1(11); M4(7); M2(7)
Job 4: M3(15); M4(10)	Job 4: M2(7); M4(8); M1(12); M2(6)
ALT1: M4(17); M2(8)	ALT1: M3(9); M2(6); M4(10); M1(8)
ALT2: M1(17); M3(8)	ALT2: M1(8); M3(9); M2(9); M4(6)
Job 5: M1(8); M2(10); M3(15); M4(17)	Job 5: M1(9); M2(7); M4(8); M2(10); M3(8)
ALT1: M3(10); M4(12); M2(13); M1(15)	ALT1: M3(8); M4(8); M3(10); M1(9); M2(7)
ALT2: M2(9); M1(8); M4(13); M3(17)	ALT2: M2(8); M3(8); M1(9); M3(8); M4(9)
	Job 6: M2(10); M3(15); M4(8); M1(15)
	ALT1: M1(8); M4(14); M2(10); M3(16)
	ALT2: M1(9); M2(16); M3(9); M4(14)

Table 5.3. Data for the Job Sets Used in Example Problems (Kumar et al., 2011) (Part2)

Job set 5	**Job set 6**
Job 1: M1(6); M2(12); M4(9)	Job 1: M1(9); M2(11); M4(7)
ALT1: M3(7); M4(10); M2(10)	ALT1: M3(10); M1(9); M2(8)
ALT2: M2(7); M3(10); M1(10)	ALT2: M4(11); M3(10); M1(6)
Job 2: M1(18); M3(6); M2(15)	Job 2: M1(19); M2(20); M4(13)
ALT1: M3(16); M2(7); M1(16)	ALT1: M4(20); M3(18); M2(14)
ALT2: M2(17); M4(8); M3(14)	ALT2: M2(18); M3(22); M1(12)
Job 3: M3(9); M4(3); M1(12)	Job 3: M2(14); M3(20); M4(9)
ALT1: M1(11); M3(4); M2(9)	ALT1: M3(12); M4(21); M1(10)
ALT2: M4(10); M2(5); M3(11)	ALT2: M4(15); M1(21); M2(7)
Job 4: M4(6); M2(15)	Job 4: M2(14); M3(20); M4(9)
ALT1: M2(8); M3(13)	ALT1: M4(13); M1(19); M3(11)
ALT2: M3(7); M1(14)	ALT2: M3(12); M4(19); M1(12)
Job 5: M3(3); M1(9)	Job 5: M1(11); M3(16); M4(8)
ALT1: M4(5); M2(7)	ALT1: M2(12); M1(14); M3(9)
ALT2: M2(4); M4(8)	ALT2: M3(12); M4(13); M2(10)
	Job 6: M1(10); M3(12); M4(10)
	ALT1: M3(9); M4(11); M2(12)
	ALT2: M2(9); M1(11); M3(12)
Job set 7	**Job set 8**
Job 1: M1(6); M4(6)	Job 1: M2(12); M3(21); M4(11)
ALT1: M3(7); M2(5)	ALT1: M1(13); M4(19); M3(12)
ALT2: M4(5); M3(7)	ALT2: M3(14); M2(18); M1(12)
Job 2: M2(11); M4(9)	Job 2: M2(12); M3(21); M4(11)
ALT1: M4(10); M3(10)	ALT1: M3(14); M2(20); M1(10)
ALT2: M3(12); M1(8)	ALT2: M1(13); M4(19); M3(12)
Job 3: M2(9); M4(7)	Job 3: M2(12); M3(21); M4(11)
ALT1: M3(10); M1(6)	ALT1: M4(13); M1(22); M3(9)
ALT2: M4(8); M3(8)	ALT2: M1(10); M4(22); M2(12)
Job 4: M3(16); M4(7)	Job 4: M2(12); M3(21); M4(11)
ALT1: M2(14); M3(9)	ALT1: M3(11); M1(23); M2(10)
ALT2: M1(17); M2(6)	ALT2: M4(13); M2(22); M3(9)
Job 5: M1(9); M3(18)	Job 5: M1(10); M2(14); M3(18); M4(9)
ALT1: M4(11); M2(20)	ALT1: M4(13); M3(13); M1(17); M2(8)
ALT2: M2(10); M4(17)	ALT2: M3(11); M4(13); M2(16); M1(11)
Job 6: M2(13); M3(19); M4(6)	Job 6: M1(10); M2(14); M3(18); M4(9)
ALT1: M3(12); M4(18); M1(8)	ALT1: M2(12); M4(13); M1(16); M3(10)
ALT2: M1(14); M2(17); M3(7)	ALT2: M4(9); M3(15); M2(17); M1(10)
	Job 7: M1(10); M2(9); M3(13)
	ALT1: M3(11); M1(10); M2(11)
	ALT2: M4(11); M3(10); M1(11)
	Job 8: M1(11); M2(9); M4(8)
	ALT1: M2(9); M3(10); M1(9)
	ALT2: M4(10); M1(11); M3(7)

Table 5.4. Data for the Job Sets Used in Example Problems (Kumar et al., 2011) (Part3)

Job set 9	**Job set 10**
Job 1: M3(9); M1(12); M2(9); M4(6)	Job 1: M1(11); M3(19); M2(16); M4(13)
ALT1: M4(10); M3(9); M1(10); M2(7)	ALT1: M2(13); M4(17); M3(14); M1(15)
ALT2: M2(8); M4(15); M3(8); M1(5)	ALT2: M4(13); M2(18); M1(14); M3(14)
Job 2: M3(16); M2(11); M4(9)	Job 2: M2(21); M3(16); M4(14)
ALT1: M1(14); M3(12); M2(10)	ALT1: M3(19); M4(17); M1(15)
ALT2: M2(18); M4(10); M3(8)	ALT2: M1(20); M4(18); M3(13)
Job 3: M1(21); M2(18); M4(7)	Job 3: M3(8); M2(10); M1(14); M4(9)
ALT1: M3(19); M4(19); M1(8)	ALT1: M2(10); M4(8); M3(16); M1(7)
ALT2: M4(22); M1(16); M2(8)	ALT2: M1(10); M4(8); M2(15); M3(8)
Job 4: M2(20); M3(22); M4(11)	Job 4: M2(13); M3(20); M4(10)
ALT1: M4(21); M1(20); M3(12)	ALT1: M1(15); M4(17); M3(11)
ALT2: M3(22); M2(21); M1(10)	ALT2: M3(11); M1(20); M2(12)
Job 5: M3(14); M1(16); M2(13); M4(9)	Job 5: M1(9); M3(16); M4(18)
ALT1: M2(15); M4(13); M1(14); M3(10)	ALT1: M3(12); M4(14); M1(17)
ALT2: M1(16); M3(14); M4(11); M2(11)	ALT2: M4(10); M1(17); M2(16)
	Job 6: M2(19); M1(21); M3(11); M4(15)
	ALT1: M1(16); M3(22); M4(14); M2(14)
	ALT2: M4(17); M2(19); M1(13); M3(17)

For experimental design of the problem, three factors are determined. These are; machine layout (4 Levels), job sets (10 Levels) and the relative importance of travelling times to processing times (2 Levels). In addition, there are four machines and two AGVs are available in the current system. Ten different job sets with different job sequencings and processing times are generated and they are submitted in the definition of the problem. By means of different combinations of these ten job sets and four layout plans, 82 test problems are generated. In Table 5.5 and Table 5.6, the performance results of the proposed FMAS are seen with result of the other methods for t/p ratio >0.25.

Table 5.5. Comparison of the FMAS with Offline Methods (for t/p ratio>0.25)(Part1)

Example	STW (Bilge 1995)	UGA (Ulusoy 1997)	AGA (Abdelmauid 2004)	PGA (Reddy 2006)	DE (Babu 2010)	MAS (Erol 2012)	FDE (Kumar 2011)	FMAS (Proposed)
EX11	96	96	96	96	94	130	96	111
EX21	105	104	102	100	108	143	100	128
EX31	105	105	99	99	87	142	99	114
EX41	118	116	112	112	85	198	112	163
EX51	89	87	87	87	80	130	87	97
EX61	120	121	118	118	114	153	118	138
EX71	119	118	115	111	90	129	114	124
EX81	161	152	161	161	145	196	161	191
EX91	120	117	118	116	115	178	114	138
EX101	153	150	147	147	121	188	147	170
EX12	82	82	82	82	88	98	82	87
EX22	80	76	76	76	86	86	76	88
EX32	88	85	85	85	74	114	85	99
EX42	93	88	88	87	74	129	86	99
EX52	69	69	69	69	76	98	69	78
EX62	**100**	**98**	**98**	**98**	**92**	**123**	**98**	**94**
EX72	**90**	**85**	**79**	**79**	**70**	**92**	**79**	**78**
EX82	**151**	**142**	**151**	**151**	**123**	**172**	**151**	**149**
EX92	104	102	104	102	95	123	104	107
EX102	139	137	136	135	113	154	135	154

STW, Sliding Time Window; UGA, Ulusoy Genetic Algorithm; AGA, Adaptive Genetic Algorithm; PGA, Proposed Genetic Algorithm; DE, Differential Evolution Algorithm, MAS, Multi-Agent System; FDE Flexible Differential Evolution, FMAS Flexible Multi-Agent System.

According to the results, the proposed multi agent system (FMAS) is developed as taking into account machine flexibility and FMAS gives better results in the 37 of 40 instances than the MAS model having no machine flexibility. In addition, it dominates other offline algorithms in examples seen as bold in Table 5.5 and Table 5.6.

Table 5.6. Comparison of the FMAS with Offline Methods (for t/p ratio>0.25)(Part2)

Example	STW (Bilge 1995)	UGA (Ulusoy 1997)	AGA (Abdelmauid 2004)	PGA (Reddy 2006)	DE (Babu 2010)	MAS (Erol 2012)	FDE (Kumar 2011)	FMAS (Proposed)
EX13	84	84	84	84	82	109	84	91
EX23	86	86	86	86	100	98	86	102
EX33	86	86	86	86	78	103	86	102
EX43	95	91	89	89	69	155	89	123
EX53	76	75	74	74	72	109	74	88
EX63	104	104	104	103	101	128	103	113
EX73	91	88	86	83	80	93	84	95
EX83	**153**	**143**	**153**	**153**	**137**	**172**	**153**	**146**
EX93	110	105	106	105	104	119	103	107
EX103	143	143	141	139	119	158	139	150
EX14	108	103	103	103	102	168	103	128
EX24	116	113	108	108	102	169	106	131
EX34	116	113	111	111	86	167	110	128
EX44	126	126	126	126	92	242	126	168
EX54	99	97	96	96	86	168	95	115
EX64	120	123	120	120	93	189	120	144
EX74	**136**	**128**	**127**	**126**	**112**	**156**	**126**	**124**
EX84	163	163	163	163	153	251	163	194
EX94	125	123	122	122	109	181	123	138
EX104	171	164	159	158	119	246	158	180

STW, Sliding Time Window; UGA, Ulusoy Genetic Algorithm; AGA, Adaptive Genetic Algorithm; PGA, Proposed Genetic Algorithm; DE, Differential Evolution Algorithm, MAS, Multi-Agent System; FDE Flexible Differential Evolution, FMAS Flexible Multi-Agent System.

In Figure 5.2, the performance results of two problems and the FMAS's performance are seen. According to this figure, Differential Evolution Algorithm (FDE) developed by Kumar et al. (2011), compared to the first model (DE) not capturing machine flexibility developed by Babu et al. (2010), gives better results in the 7 of 40 instances. However, the proposed FMAS can generate better schedules for new problems and increase its performance to higher levels.

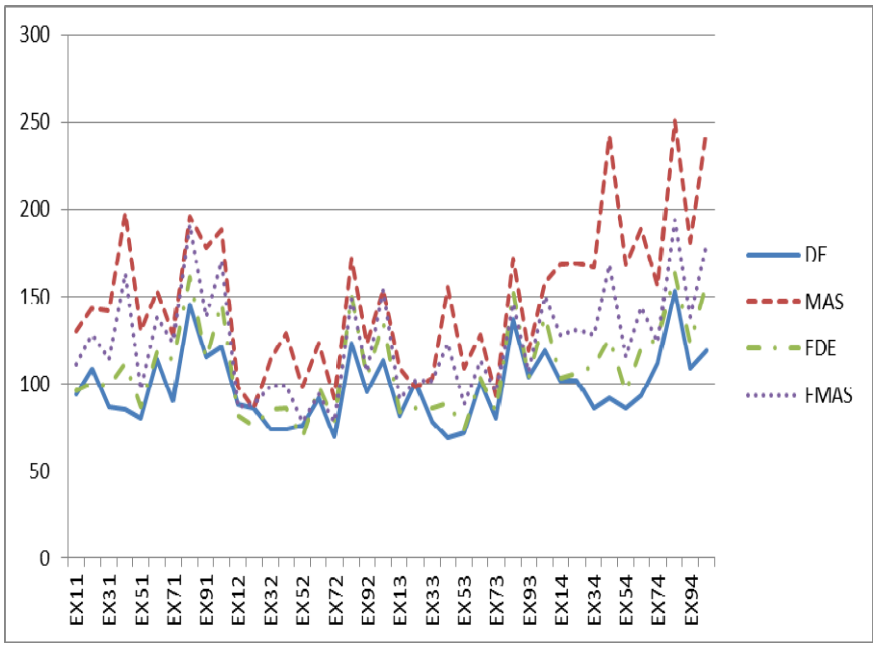


Figure 5.2. Comparison between Flexible and Non-Flexible Methods (for t/p ratio > 0.25)

As seen in Figure 5.3, FDE's results are usually better than FMAS's results. However, most of the results of both approaches are close to each other.

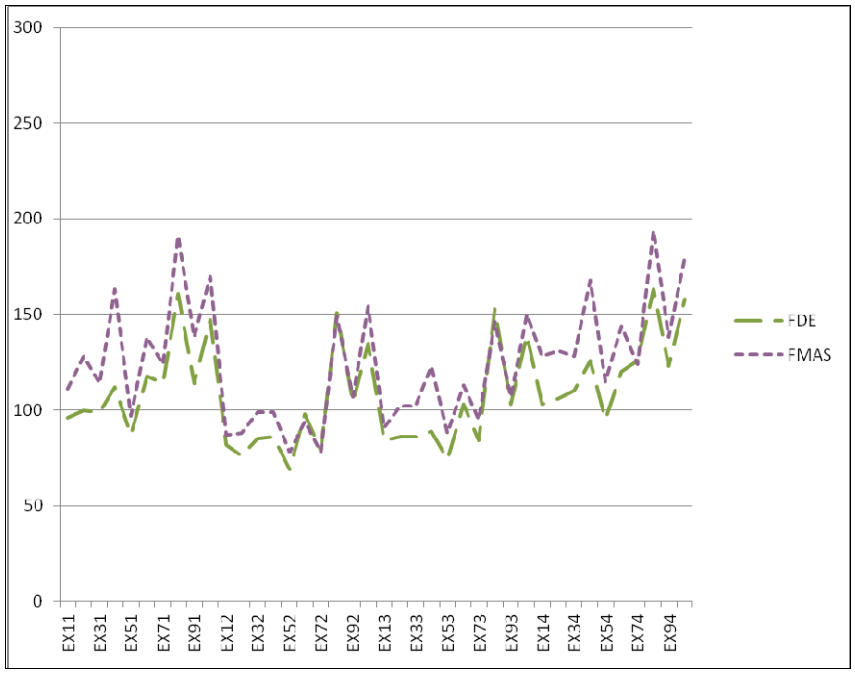


Figure 5.3. Comparison between FDE and FMAS (for t/p ratio > 0.25)

According to the results, in terms of total production time, the offline algorithms perform better than FMAS. Although FMAS is not optimization algorithm and designed as online, it is possible to generate schedule to approximate those generated by offline algorithms.

Another analysis is made for the lower rate. These results are given in Table 5.7 and Table 5.8. In these tables, another digit was added to instance code. 0 or 1 was written at the end of code showing that travelling times decreased half as much as the previous one and also as for two case processing times, they are doubled and tripled respectively. For example, EX110 shows that the first job took the first layout plan (Figure 5.1) and doubled processing periods (Table 5.2, Table 5.3, and Table 5.4).

Table 5.7. Comparison of the FMAS with Offline Methods (for t/p ratio<0.25)(Part1)

Example	STW (Bilge 1995)	UGA (Ulusoy 1997)	AGA (Abdelmauid 2004)	PGA (Reddy 2006)	DE (Babu 2010)	MAS (Erol 2012)	FDE (Kumar 2011)	FMAS (Proposed)
EX110	126	126	126	126	133	135	126	147
EX210	**148**	**148**	**148**	**148**	**159**	**157**	**148**	**124**
EX310	150	148	150	150	133	154	150	159
EX410	121	119	119	119	102	211	119	171
EX510	102	102	102	102	109	118	102	115
EX610	**186**	**186**	**186**	**186**	**169**	**204**	**186**	**174**
EX710	137	137	137	137	118	138	137	139
EX810	**292**	**271**	**292**	**292**	**233**	**330**	**292**	**217**
EX910	**176**	**176**	**176**	**176**	**162**	**191**	**176**	**174**
EX1010	**238**	**236**	**238**	**238**	**184**	**269**	**238**	**221**
EX120	123	123	123	123	134	127	123	141
EX220	**143**	**143**	**143**	**143**	**148**	**151**	**143**	**127**
EX320	148	145	145	145	126	144	145	145
EX420	116	114	114	114	100	161	114	176
EX520	100	100	100	100	110	110	100	101
EX620	**183**	**181**	**181**	**181**	**116**	**196**	**181**	**160**
EX720	**136**	**136**	**136**	**136**	**113**	**132**	**136**	**113**
EX820	**287**	**268**	**287**	**287**	**222**	**319**	**287**	**192**
EX920	174	173	173	173	156	187	173	188
EX1020	236	238	236	236	173	266	236	240

STW, Sliding Time Window; UGA, Ulusoy Genetic Algorithm; AGA, Adaptive Genetic Algorithm; PGA, Proposed Genetic Algorithm; DE, Differential Evolution Algorithm, MAS, Multi-Agent System; FDE Flexible Differential Evolution, FMAS Flexible Multi-Agent System.

According to the results obtained from Table 5.7 and Table 5.8, in the problems which have comparatively lower t/p ratio, the relative performance of FMAS is much better. In almost all the problems, the proposed FMAS achives a reasonable success regarding schedules generated by offline methods. In 22 of 42 instances FMAS dominates offline algorithms. The results indicate that FMAS gives better performance via decreasing t/p ratios.

Table 5.8. Comparison of the FMAS with Offline Methods (for t/p ratio<0.25)(Part2)

Example	STW (Bilge 1995)	UGA (Ulusoy 1997)	AGA (Abdelmauid 2004)	PGA (Reddy 2006)	DE (Babu 2010)	MAS (Erol 2012)	FDE (Kumar 2011)	FMAS (Proposed)
EX130	122	122	122	122	129	134	122	159
EX230	**146**	**146**	**146**	**146**	**155**	**151**	**146**	**138**
EX330	**149**	**146**	**146**	**146**	**127**	**129**	**151**	**137**
EX430	116	114	114	114	99	228	114	154
EX530	**99**	**99**	**99**	**99**	**105**	**111**	**99**	**92**
EX630	**184**	**182**	**182**	**182**	**165**	**198**	**182**	**162**
EX730	**137**	**137**	**137**	**137**	**118**	**132**	**136**	**133**
EX830	**288**	**270**	**288**	**288**	**229**	**273**	**288**	**226**
EX930	176	174	174	174	158	187	174	179
EX1030	**237**	**241**	**237**	**237**	**182**	**266**	**237**	**196**
EX140	124	124	124	124	133	137	124	145
EX241	217	217	217	217	126	230	217	220
EX340	**151**	**151**	**151**	**151**	**133**	**155**	**151**	**140**
EX341	**222**	**221**	**221**	**221**	**161**	**227**	**221**	**215**
EX441	179	172	172	172	151	344	171	256
EX541	154	148	148	148	177	158	148	180
EX640	**185**	**184**	**184**	**184**	**170**	**211**	**184**	**180**
EX740	138	137	137	137	118	158	137	138
EX741	**203**	**203**	**203**	**203**	**171**	**206**	**203**	**149**
EX840	**293**	**273**	**293**	**293**	**237**	**331**	**293**	**232**
EX940	177	175	175	175	159	195	175	183
EX1040	**240**	**244**	**240**	**240**	**184**	**276**	**240**	**228**

STW, Sliding Time Window; UGA, Ulusoy Genetic Algorithm; AGA, Adaptive Genetic Algorithm; PGA, Proposed Genetic Algorithm; DE, Differential Evolution Algorithm, MAS, Multi-Agent System; FDE Flexible Differential Evolution, FMAS Flexible Multi-Agent System.

For the problem sets, with the decreasing t/p values, the performances of DE and MAS approaches are compared to FDE and FMAS approaches. As it can be seen

in Table 5.7, Table 5.8 and Figure 5.4, FMAS approach achieves to generate better schedules than MAS approaches for 30 instances out of 42 samples. However, the same achievement can not be realized for FDE approach versus DE approach.

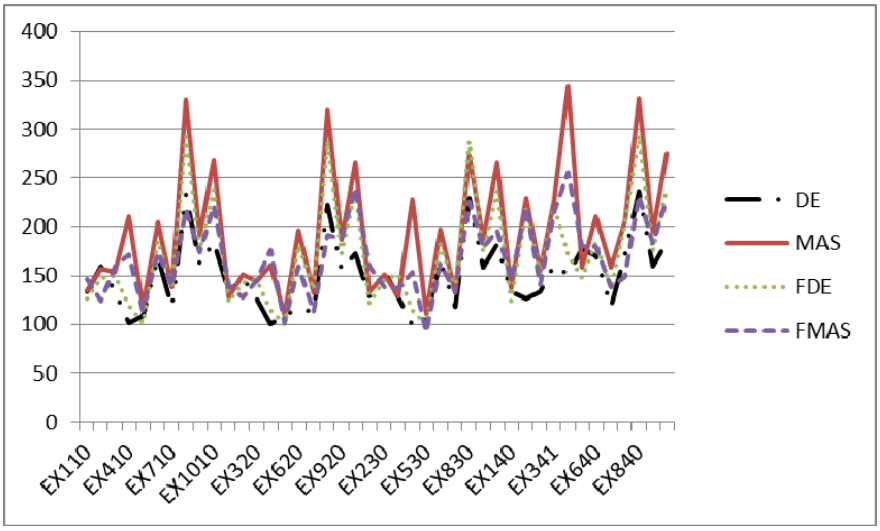


Figure 5.4. Comparison between Flexible and Non-Flexible Methods (for t/p ratio <0.25)

Flexible Differential Evolution Algorithm and Flexible Multi Agent System are depicted separately in Figure 5.5. In 22 of 42 instances, FMAS yields better makespan minimization results than those in FDE.

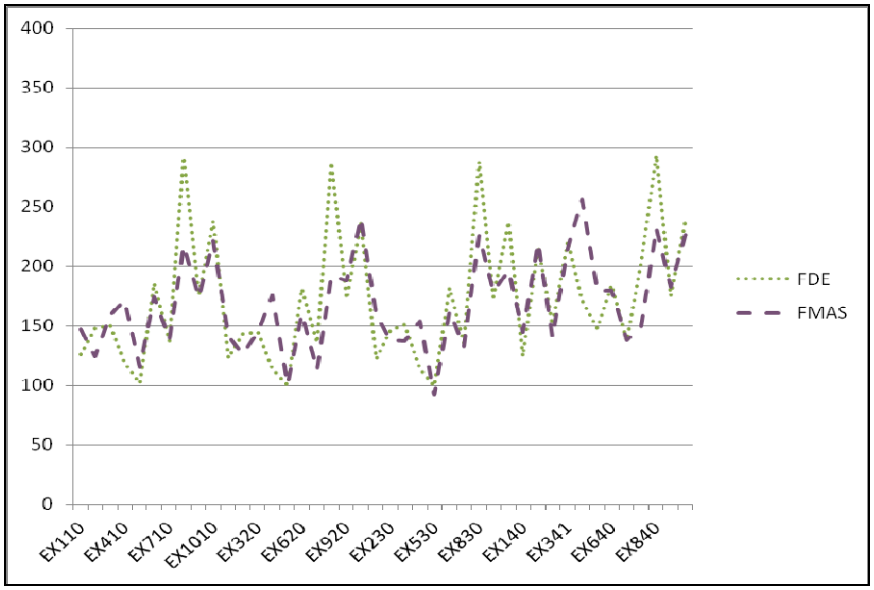


Figure 5.5. Comparison between FDE and FMAS (for t/p ratio<0.25)

In the literature related to the simultaneous scheduling of machines and AGVs, there are very few studies. Therefore, this study provides a contribution to filling this gap. The proposed FMAS system yields better results in the majority of the simulation tests. These results suggest that the proposed FMAS has a promising performance in simultaneous scheduling of flexible machines and AGVs and it also can be developed better.

The system could update its state by removing or adding resources. Percepts update the status of resources as available or unavailable. As seen in Figure 5.6, AGV3 could enter the system after the system starts up.

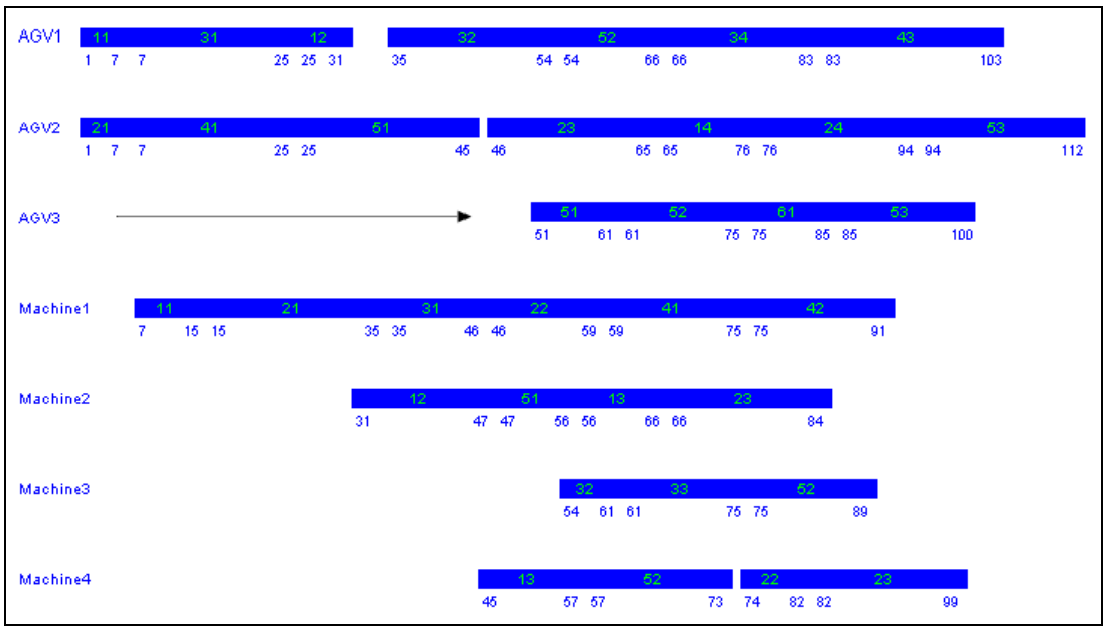


Figure 5.6. The Gantt chart of Including New AGVs

By means of its negotiating and bidding mechanism, the proposed system proves its adaptability in generating schedules for newly arrived jobs in a dynamic fashion.

One of the best advantages of the proposed system is that thanks to the negotiation and bidding mechanisms, more dynamic schedules can be obtained when the new jobs enter the system. On the other hand, the use of offline algorithms can not permit to generate dynamic schedules.

In conclusion, the proposed system inherently includes proper operation decisions for machine and vehicle assignments in a dynamic environment. The system operates orders by dynamically regardless of the time; this dynamic structure can inspire future researchers so that they will be able to obtain better results.

6. CONCLUSIONS AND FUTURE WORKS

6.1. Summary of the Study

Up to the present, the classical scheduling approaches which ignore the dynamic structure of the manufacturing environment are used mostly to find an optimal solution; therefore, their solutions are far away from real-world applications. Multi-Agent Systems are the systems that take the dynamic structure of manufacturing environment into account and give realistic solutions. The agents in the system can interact with each other in a cooperative fashion for resources or feedback to accomplish the task. This procedure saves time and offers a multidisciplinary solution, which would not have been possible with a stand-alone system. Moreover, agents can also be used for problems that can be solved based on other methodologies, but in a significantly better and more natural way.

MASs can be used as online scheduling methods and obtained a better scheduling performance. MASs also have flexibility, reliability, adaptability, and reconfigurability. A decentralized multi-agent system is based on the idea that several distributed decision-makers (or agents) can cooperate and coordinate in order to obtain globally optimal performances. Decentralized nature of multi-agent systems can also be a benefit to dynamic manufacturing scheduling as, only directly affected agents become involved and not other agents.

The framework for the proposed architecture is composed through Prometheus^{TM} methodology; thus MAS is used to simulate it. The proposed system is compared with the several test problems from the literature. Ten different job sets with different processing times, and part placements in problem definition and four layouts are used. The examples are classified as two problems; whose t/p ratios are higher than 0.25 and those with lower t/p ratios. According to 40 examples for t/p ratio>0.25, FMAS can obtain better schedules with the success of 92.5% when compared to FMAS. On the other hand, FMAS can dominate MAS with the success of 71% for t/p ratio<0.25. The proposed FMAS with integrated flexibility of

machines has the ability to get optimal or better results than some of instances in the literature.

6.2. Conclusions

In this study, a Flexible Multi-Agent System is proposed for simultaneous scheduling of vehicles and flexible machines in Flexible Manufacturing Systems. There are very few studies related to the application of FMAS to dynamic manufacturing planning and control problems. Decentralized control reduces the computational complexity, so the proposed FMAS follows optimal assignment of process by means of alternative routes for machines.

In conclusion, the results and experiences obtained in this research have shown that there are really big potential and performance improvement possibilities by the proper implementation of FMAS systems in manufacturing. The results obtained in this thesis will be encouring to the next researchers to get better solutions.

6.3. Future Directions

After completion of this thesis, there are still some gaps which are requires an extensive persperctive. These gaps suggest directions for the future researchers:

- Further studies are needed to adopt unexpected events such as resource breakdowns that can occur in a manufacturing system.
- The AGV system could carry more load types instead of one type by trying to combine the most appropriate load charts.
- A multi agents system could be extended to deal with the some problems such as the traffic control, the machine failure or downtime, or the vehicle dispatches for battery changer.
- In order to obtain a better system performance, agent based system could be designed by JACK^{TM} team module by means of social agents.

REFERENCES

ABDELMAGUID, T. F., A. O. NASSEF, et al., 2004. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. International Journal of Production Research, 42(2): 267-281.

AOS, 2009 JACK^{TM} Intelligent Agents Agent Manuel. Carlton South, Victoria, AUSTRALIA, Agent Orianted Software Pty. Ltd.

BABU, A. G., J. JERALD, et al., 2010. Scheduling of machines and automated guided vehicles in FMS using differential evolution. International Journal of Production Research, 48(16): 4683-4699.

BAYKASOGLU, A., V. KAPLANOGLU, et al., 2011. A multi-agent framework for load consolidation in logistics. Transport, 26(3): 320-328.

BAYKASOGLU, A. AND L. ÖZBAKIR, 2008. Analysing the effect of flexibility on manufacturing systems performance. Journal of Manufacturing Technology Management, 19(2): 172-193.

BAYKASOGLU, A. AND L. ÖZBAKIR, 2010. Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system. International Journal of Production Economics, 124(2): 369-381.

BAYKASOĞLU, A. ,2009. Quantifying machine flexibility. International Journal of Production Research, 47(15): 4109-4123.

BAYKASOĞLU, A. AND Z. D. U. DURMUŞOĞLU, 2012. Flow time analyses of a simulated flexible job shop by considering jockeying. The International Journal of Advanced Manufacturing Technology, 58(5): 693-707.

BAYKASOĞLU, A., L. ÖZBAKIR, et al. ,2004. Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems. Journal of Intelligent Manufacturing, 15(6): 777-785.

BILGE, Ü. AND G. ULUSOY, 1995. A time window approach to simultaneous scheduling of machines and material handling system in an FMS. Operations Research, 43(6): 1058-1070.

BLAZEWICZ, J., K. ECKER, et al., 2007. Handbook on Scheduling: Models and Methods for Advanced Planning (International Handbooks on Information Systems), Springer-Verlag New York, Inc.

BROWNE, J., D. DUBOIS, et al., 1984. Classification of flexible manufacturing systems. The FMS magazine, 2(2): 114-117.

CHANG, A. Y., 2009. "An attribute approach to the measurement of machine-group flexibility." European Journal of Operational Research 194(3): 774-786.

CHANG, A. Y., D. J. WHITEHOUSE, et al., 2001. An approach to the measurement of single-machine flexibility. International Journal of Production Research, 39(8): 1589-1601.

DE TONI, A. AND S. TONCHIA, 1998. Manufacturing flexibility: a literature review. International Journal of Production Research, 36(6): 1587-1617.

DUFFIE, N. A., R. CHITTURI, et al., 1988. Fault-tolerant heterarchical control of heterogeneous manufacturing system entities. Journal of Manufacturing Systems, 7(4): 315-328.

EGBELU, P. J. AND J. M. A. TANCHOCO, 1984. Characterization of automatic guided vehicle dispatching rules. The International Journal of Production Research, 22(3): 359-374.

EROL, R., C. SAHIN, et al., 2012. A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems. Applied Soft Computing.

EVERTSZ, R., M. FLETCHER, et al., 2004. Implementing industrial multi-agent systems using JACK^{TM} tm. Programming multi-agent systems: 18-48.

FARAHVASH, P. AND T. O. BOUCHER, 2004. A multi-agent architecture for control of AGV systems. Robotics and Computer-Integrated Manufacturing 20(6): 473-483.

GAO, J., M. GEN, et al., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computers & Industrial Engineering, 53(1): 149-162.

HOWDEN, N., R. RÖNNQUIST, et al., 2001. JACK^{TM} intelligent agents-summary of an agent infrastructure. 5th International conference on autonomous agents.

JERALD, J., P. ASOKAN, et al., 2006. Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic algorithm. The International Journal of Advanced Manufacturing Technology, 29(5): 584-589.

KOMMA, V. R., P. K. JAIN, et al., 2011. An approach for agent modeling in manufacturing on JADE™ reactive architecture. The International Journal of Advanced Manufacturing Technology, 52(9): 1079-1090.

KUMAR, M. V. S., R. JANARDHANA, et al., 2011. Simultaneous scheduling of machines and vehicles in an FMS environment with alternative routing. The International Journal of Advanced Manufacturing Technology, 53 1(4): 339-351.

LUN, M. AND F. CHEN, 2000. Holonic concept based methodology for part routeing on flexible manufacturing systems. The International Journal of Advanced Manufacturing Technology, 16(7): 483-490.

MOHAMED, Z. M., M. A. YOUSSEF, et al., 2001. The impact of machine flexibility on the performance of flexible manufacturing systems. International Journal of Operations & Production Management 21(5/6): 707-742.

PADGHAM, L. AND M. WINIKOFF, 2004. Developing intelligent agent systems, John Wiley.

REDDY, B. S. P. AND C. S. P. RAO, 2006. A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS. The International Journal of Advanced Manufacturing Technology, 31(5): 602-613.

SABUNCUOGLU, I. AND D. HOMMERTZHEIM, 1989. An investigation of machine and AGV scheduling rules in an FMS. Proceedings of the third ORSA/TIMS conference on flexible manufacturing Systems: Operations Research Models and Applications.

SABUNCUOGLU, I. AND D. L. HOMMERTZHEIM, 1992. Dynamic dispatching algorithm for scheduling machines and automated guided vehicles in a flexible manufacturing system. The International Journal of Production Research, 30(5): 1059-1079.

SABUNCUOGLU, I. AND D. L. HOMMERTZHEIM, 1992. Experimental investigation of FMS machine and AGV scheduling rules against the mean flow-time criterion.The International Journal of Production Research, 30(7): 1617-1635.

SABUNCUOGLU, I. AND D. L. HOMMERTZHEIM, 1993. Experimental investigation of an FMS due-date scheduling problem: Evaluation of machine and AGV scheduling rules. International Journal of Flexible Manufacturing Systems, 5(4): 301-323.

SCOTT, W. D. 2004. A flexible control system for flexible manufacturing systems, Texas A&M University.

SETHI, A. K. AND S. P. SETHI, 1990. Flexibility in manufacturing: a survey. International Journal of Flexible Manufacturing Systems, 2(4): 289-328.

SRIVASTAVA, S. C., A. K. CHOUDHARY, et al., 2008. Development of an intelligent agent-based AGV controller for a flexible manufacturing system. The International Journal of Advanced Manufacturing Technology 36(7): 780-797.

SUBBAIAH, K., M. N. RAO, et al., 2009. Scheduling of AGVs and machines in FMS with makespan criteria using sheep flock heredity algorithm. International Journal of Physical Sciences, 4(2): 139-148.

SUBRAMANIAM, V., G. LEE, et al., 2000. Machine selection rules in a dynamic job shop. The International Journal of Advanced Manufacturing Technology 16(12): 902-908.

SYCARA, K. P., 1998. Multiagent systems. AI magazine, 19(2): 79.

ŞAHİN, C., 2010. Multi-agent approach for the scheduling of manufacturing systems. Industrial Engineering. Adana, Çukurova Üniversitesi. Ph.D.: 116.

TANAKA, Y., T. NISHI, et al., 2010. Dynamic Optimization of Simultaneous Dispatching and Conflict-free Routing for Automated Guided Vehicles. Journal of Advanced Mechanical Design, Systems, and Manufacturing 4(3): 701-715.

ULUSOY, G., F. SIVRIKAYA-ŞERIFOĞLU,et al., 1997. A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles. Computers & Operations Research, 24(4): 335-351.

VINOD, V. AND R. SRIDHARAN, 2011. Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system. International Journal of Production Economics 129(1): 127-146.

WAHAB, M. I. M., D. WU, et al., 2008. A generic approach to measuring the machine flexibility of manufacturing systems. European Journal of Operational Research, 186(1): 137-149.

WALLACE, A., 2007. Multi-agent negotiation strategies utilizing heuristics for the flow of AGVs. International Journal of Production Research, 45(2): 309-322.

WOOLDRIDGE, M. AND N. R. JENNINGS, 1995. Intelligent agents: Theory and practice. Knowledge Engineering Review, 10(2): 115-152.

http://www.cs.rmit.edu.au/agents/pdt/

BIOGRAPHY

Melek DEMİRTAŞ received the B.S. degree in Industrial Engineering from Cukurova University in 2010. She started her MSc studies in Industrial Engineering Department at Cukurova University in February 2011. She worked as a postgraduate worker in a scientific project supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK), between October 2011 and September 2013 under the direction of Prof. Dr. Rızvan EROL.

APPENDIX

APPENDIX A- *Manage_and_Evaluate_Machine_Process* plan of Operation Agent

```
/*********************************************************************
 * Created by the JDE: Thu May 02 07:27:32 EEST 2013
 * WARNING: If you are editing this file from outside the JDE, you
 * should make sure to load the changes back into the JDE by setting
 * the preference "Check for outside changes..." in the "Text Editor"
 * preference tab or reload the component manually.
 * Otherwise you will lose your changes next time the file is generated.
 *********************************************************************/
package siparis;
import files.Fgiris;
import files.Mesafeler21;
import files.agvbilgi;
import files.giris;
import java.text.SimpleDateFormat;
import java.util.*;
/**
 *
 */
public plan MakineBelirle extends Plan {
#handles event MakSecim2 ev;
#posts event Makineilehaberles ev2;
#sends event MakBlackbordKontrol makbilgi;
#uses data SiparisAjaniVeriTabani data;
#uses data sistemdekiagvler agvdurum;
#uses data SiparisAjaniVeriTabani2 data2;
static boolean relevant(MakSecim2 ev)
{
return true;
```

```
}
context()
{
true;
}
#reasoning method
body()
{
Fgiris aktar2;
aktar2=ev.siparisbilgileri55;
logical long EnErkenBaslamaZamani;
logical int ffrom,to;
int MakinaUzakliklari[][];
MakinaUzakliklari=new int[3][4];
double MakinaDegerleri [][];
MakinaDegerleri=new double [3][2];
@waitFor(data2.sorgula(ev.siparisbilgileri55.siparisAdi,ev.siparisbilgileri55.operasy
onNo,ffrom,to,1,0,EnErkenBaslamaZamani));
int max=0,min=10000;
int MinMakAdi,MinOpeSuresi,MaxOpeSuresi,MaxMakAdi;
int from=ffrom.as_int();
MakinaUzakliklari[0][0]=ev.siparisbilgileri55.makinaAdi1;
MakinaUzakliklari[0][2]=Mesafeler21.mesafe2(ffrom.as_int(),ev.siparisbilgileri55.m
akinaAdi1);
MakinaUzakliklari[0][1]=ev.siparisbilgileri55.operasyonSuresi1;
MakinaUzakliklari[1][0]= ev.siparisbilgileri55.makinaAdi2;

MakinaUzakliklari[1][2]=Mesafeler21.mesafe2(ffrom.as_int(),ev.siparisbilgileri55.m
akinaAdi2);
MakinaUzakliklari[1][1]=ev.siparisbilgileri55.operasyonSuresi2;
MakinaUzakliklari[2][0]= ev.siparisbilgileri55.makinaAdi3;
```

```
MakinaUzakliklari[2][2]=Mesafeler21.mesafe2(ffrom.as_int(),ev.siparisbilgileri55.m
akinaAdi3);
MakinaUzakliklari[2][1]=ev.siparisbilgileri55.operasyonSuresi3;
for(int i=0;i<=2;i++)
{
if(MakinaUzakliklari[i][0]!=0)
{
String makadi=Integer.toString(MakinaUzakliklari[i][0]);
MakBlackbordKontrol q = makbilgi.bosmesaj2();
@send(makadi, q);
@waitFor(q.replied());
Cevapla fev = (Cevapla) q.getReply();
System.out.println(ev.siparisbilgileri55.siparisAdi+""+ev.siparisbilgileri55.operasyo
nNo+" No luis icin Makina Adi:"+MakinaUzakliklari[i][0]+ "Kuyruk
uzunlugu:"+fev.kuyrukuzunlugu);
if(from==MakinaUzakliklari[i][0])
MakinaUzakliklari[i][3]=fev.kuyrukuzunlugu;
else
MakinaUzakliklari[i][3]=fev.kuyrukuzunlugu+1;
}
else
MakinaUzakliklari[i][3]=1;
}
for(int i=0;i<=2;i++)
{
if(MakinaUzakliklari[i][1]+MakinaUzakliklari[i][2]<min)
{
min=MakinaUzakliklari[i][1]+MakinaUzakliklari[i][2];
MinMakAdi=MakinaUzakliklari[i][0];
MinOpeSuresi=MakinaUzakliklari[i][1];
}
```

```
if(MakinaUzakliklari[i][1]+MakinaUzakliklari[i][2]>max)
{
max=MakinaUzakliklari[i][1]+MakinaUzakliklari[i][2];
MaxOpeSuresi=MakinaUzakliklari[i][1];
MaxMakAdi=MakinaUzakliklari[i][0];
}
}
System.out.println(ev.siparisbilgileri55.siparisAdi+""+ev.siparisbilgileri55.operasyo
nNo+" nolu is icin Makine Secim Sonucu:");
System.out.println("Min Mak ADI:"+MinMakAdi);
System.out.println("Min Mak Ope Suresi:"+MinOpeSuresi);
System.out.println("EnErkenBalama Zamani:"+EnErkenBaslamaZamani.as_long());
if(MinOpeSuresi!=0)
{double formul,y;
MakinaDegerleri[0][0]=ev.siparisbilgileri55.makinaAdi1;
formul=((MakinaUzakliklari[0][1]+MakinaUzakliklari[0][2]-min));
formul=formul/(max-min);
y=(double)1/(MakinaUzakliklari[0][3]+1);
MakinaDegerleri[0][1]=formul+(1-y);
MakinaDegerleri[1][0]=ev.siparisbilgileri55.makinaAdi2;
formul=((MakinaUzakliklari[1][1]+MakinaUzakliklari[1][2]-min));
formul=formul/(max-min);
y=(double)1/(MakinaUzakliklari[1][3]+1);
MakinaDegerleri[1][1]=formul+(1-y);
MakinaDegerleri[2][0]=ev.siparisbilgileri55.makinaAdi3;
formul=((MakinaUzakliklari[2][1]+MakinaUzakliklari[2][2]-min));
formul=formul/(max-min);
y=(double)1/(MakinaUzakliklari[2][3]+1);
MakinaDegerleri[2][1]=formul+(1-y);
}
double mindeger=10000;
```

```
for(int kk=0;kk<=2;kk++)
{
if(MakinaDegerleri[kk][1]<mindeger)
{
mindeger=MakinaDegerleri[kk][1];
MinMakAdi=MakinaUzakliklari[kk][0];
MinOpeSuresi=MakinaUzakliklari[kk][1];
System.out.println(ev.siparisbilgileri55.siparisAdi+""+ev.siparisbilgileri55.operasyo
nNo+" nolu is icin Makine Secim Sonucu2:");
System.out.println("Min Mak ADI:"+MinMakAdi);
System.out.println("Min Mak Ope Suresi:"+MinOpeSuresi);
System.out.println("EnErkenBalama Zamani:"+EnErkenBaslamaZamani.as_long());
System.out.println("Min Deger:"+mindeger);
System.out.println("k degeri:"+kk);
} }
System.out.println(ev.siparisbilgileri55.siparisAdi+""+ev.siparisbilgileri55.operasyo
nNo+" nolu is icin Secim sonucu: From: "+from+" To: "+MinMakAdi);
data.add(ev.siparisbilgileri55.siparisAdi,ev.siparisbilgileri55.operasyonNo,MinMak
Adi,MinOpeSuresi,from,MinMakAdi,2,1,0,EnErkenBaslamaZamani.as_long());
giris                                                    aktar=new
giris(ev.siparisbilgileri55.siparisAdi,ev.siparisbilgileri55.operasyonNo,MinMakAdi,
MinOpeSuresi,from,MinMakAdi,2);
@post(ev2.request(aktar));
}
}
```

APPENDIX B- *Add_Process_Request_to_the_Blackboard* plan of Machine Agent

```
/*********************************************************************
 * Created by the JDE: Wed May 01 12:17:12 EEST 2013
 * WARNING: If you are editing this file from outside the JDE, you
 * should make sure to load the changes back into the JDE by setting
 * the preference "Check for outside changes..." in the "Text Editor"
 * preference tab or reload the component manually.
 * Otherwise you will lose your changes next time the file is generated.
 *********************************************************************/
package makine;
import files.giris;
import java.io.*;
import java.util.*;
/**
 *
 */
public plan BlackboradaYaz extends Plan {
#handles event makineyeisyolla ev;
#uses data makinablackboard data;
static boolean relevant(makineyeisyolla ev)
{
return true;
}
context()
{
true;
}
#reasoning method
```

```
body()
{
data.add(ev.siparisbilgileri6.siparisAdi,ev.siparisbilgileri6.operasyonNo,geliszamani,
ev.siparisbilgileri6.makinaAdi1,ev.siparisbilgileri6.operasyonSuresi1,ev.siparisbilgil
eri6.from,ev.siparisbilgileri6.to,0);
System.out.println(ev.siparisbilgileri6.siparisAdi+""+ev.siparisbilgileri6.operasyonN
o+" nolu is "+"Makina"+ev.siparisbilgileri6.makinaAdi1+"  Blackborduna yazildi.");
}
}
```