Universität Stuttgart
Institut für Automatisierungs- und Softwaretechnik
Prof. Dr.-Ing. Dr. h. c. P. Göhner

26.01.2005

Wahlstudienarbeit

Igor Terzi 2032 WSA

Analysis and Conception of Coordination and Collaboration Mechanisms for Agent Oriented Systems

Betreuer: Prof. Dr.-Ing. Dr. h. c. Peter Göhner
Dipl.-Ing. Hisham Mubarak, M. Sc.
Dipl.-Ing. Thomas Wagner

Abstract

Significant growth of the complexity of industrial automation systems raises the requirements to higher reliability, flexibility, scalability, adaptability and the ability to integrate these systems. The agent-oriented approach is able to fulfil these requirements because of it’s principles like decentralization, autonomy, goal orientation, reactivity and proactivity of individual agents and interaction between agents, etc. Since under agent-oriented approach the control is distributed, to act like one unit and to reach global goals the agents need to coordinate and to collaborate. The coordination and collaboration techniques should correspond to the concepts of the agent-oriented methodology and put into practice the joint activity of the agents. These techniques should also take into account the characteristics of the environment, to which the software system is applied.

The general purpose of this project is the fundamental research in the field of coordination and negotiation techniques for decentralized agent-oriented systems. For that the analysis and the general classification of already present coordination and negotiation mechanisms known in the literature were done. As far as the coordination technique is applied to a flexible production environment, this environment was described and analyzed on the example of the model process “Lego-robots”. After that it was examined which coordination mechanisms is applicable to the flexible production environment. Then, to make a coordination mechanism elaboration possible, the agent-oriented analysis of the system was carried out and the activity selection mechanism was developed. On this basis the conception for a coordination and collaboration technique for the flexible production environment automation system was finally developed and analyzed with the help of evaluation scenarios.

Kurzfassung

Die immer größere Komplexität industrieller Automatisierungssysteme stellt wachsende Anforderungen an höhere Zuverlässigkeit, Flexibilität, Skalierbarkeit, Anpassbarkeit und Integrationsfähigkeit dieser Systeme. Der agentenorientierte Ansatz kann diese Anforderungen aufgrund zugrunde liegender Prinzipien wie der Dezentralisierung, Autonomie, Ziellagebestimmung, Reaktivität und Proaktivität der individuellen Agenten und der Interaktion zwischen verschiedene Agenten erfüllen. Da beim agentenorientierten Ansatz die Steuerung verteilt ist, müssen sich Agenten abstimmen und ihre Aktivitäten koordinieren, um als eine Einheit handeln und globale Ziele erreichen. Die Koordinations- und Abstimmungsmechanismen sollten die Konzepte der agentenorientierten Methode Konzepten berücksichtigen. Diese Mechanismen sollten die Eigenschaften der Umgebung, in welcher das entwickelte Software-System eingesetzt wird, berücksichtigen.

Ziel dieses Projekts ist die grundlegende Untersuchung von Mechanismen zur Koordination und Abstimmung in Agentensystemen. Zu diesem Zweck wurde eine Analyse und Klassifikation der in der Literatur bekannten Koordinations- und Abstimmungsmechanismen durchgeführt. Da die Koordinationsmechanismen an einer flexiblen Produktionsumgebung angewendet werden sollen, wird diese Umgebungen am Beispiel des Modelprozesses "Lego-Roboter" beschrieben und analysiert. Weiter wird überprüft welche Koordinationsmechanismen für die beschriebene flexible Produktionsumgebung anwendbar sind. In einem weitern Schritt wird eine agentenorientierte Analyse des Systems durchgeführt und es wird ein Mechanismus entwickelt, der die Auswahl einer Tätigkeit unter Alternativen ermöglicht. Auf dieser Grundlage werden die Koordinations- und Abstimmungsmechanismen für die Automatisierungssysteme für flexible Produktionsumgebung entwickelt und mit der Hilfe von Evaluierungsszenarios analysiert.

Table of contents

1 Introduction

1.1 Motivation

Significant growth of complexity of industrial automation systems raises the requirements to higher reliability, flexibility, scalability, adaptability and the ability to integrate these systems. Agents and multi-agent systems represent a new approach to the development of complex software systems on the whole and industrial automation systems in particular. Because of it's principles like decentralization, autonomy, goal orientation, reactivity and proactivity of individual agents and interaction between agents, the agent-oriented approach is able to fulfil the above mentioned requirements.

Since under agent-oriented approach the control is distributed, to act like one unit and to reach global goals the agents within multi-agent systems need to collaborate and to coordinate their actions. The coordination and collaboration techniques should correspond to agent-oriented methodology concepts and put into practice the joint activity of the agents. These techniques should also take into account the characteristics of the developed software system and the environment, to which this software system is applied. Under fulfilling these conditions the coordination between agents may significantly increase the efficiency of the agents' joint activities.

So, coordination and collaboration are very important in distributed multi-agent systems. However not much research was done on the coordination and collaboration mechanisms for flexible production environments that take into consideration and apply the basic principles of agent-oriented methodology. For those reasons the coordination and collaboration mechanisms for agent-oriented systems will be analyzed and developed during in the context of this project.

1.2 Purposes of the project

The general purpose of this project is the fundamental research in the field of coordination and negotiation techniques for self-organising decentralized agent-oriented systems.

The first purpose is to make an analysis and general classification of already present coordination and negotiation mechanisms known in the literature and to examine how these mechanisms may be applied to flexible production environment automation system. For that the flexible production environment should be analyzed and described. As an example the IAS model process "Lego-robots" is used.

The second purpose of the project is, on the basis of above described analysis, to develop the conception for coordination and collaboration activities for the flexible production environment automation system on the example of process model "Lego-robots". For that the agent-oriented analysis of the system with the aim of roles and interactions models definition should be executed, and the activity selection mechanism should be developed.

1.3 Project overview

In chapter two the description and analysis of a flexible production environment will be carried out. Also the brief description of the IAS process model “Lego-robots” will be presented. After that in the third comes chapter the revision of agent-oriented approach concepts and multi-agent systems principles presented in the literature.

In the fourth chapter the analysis and classification of known coordination techniques is represented. For that it is being analyzed what reasons agent have to coordinate with each other and which properties the coordination technique to be developed should have. Next follows the classification of possible coordination techniques.
On the basis of chapter four it is analyzed, how known coordination techniques may be applied to a flexible production environment. Particularly four approaches are marked out. This information may be found in chapter five.

Then the development process actually begins. First, the agent-oriented analysis of the system is carried out in chapter six, with the aim to develop roles and interactions models. The next important point of the work is the development of the activity selection mechanism that allows agents pursue several goals simultaneously and to execute a global task jointly. Information about that is represented in chapter seven.

On the basis of above mentioned analysis and development result the coordination technique for flexible production environment “Lego-robots” is developed. Particularly, first the application of different coordination approaches is analyzed, and then the coordination mechanism is developed. This development is described in chapter eight.

The last point of the work is the elaboration of evaluation scenarios. The aim of that is to explain how the system works under different coordination approaches and to show that the developed coordination technique gives the best result.

2 Description and Analysis of the Developed Production Environment

The task environment, as it is said in [RuNo03], are the problems to which agent- and multi-agent systems are the solution. So, the characteristics of the environment directly affect the way the agent-program should be developed.

In general, task environment is composed of:
- actually *environment*: the environment in which the agent is running.
- *performance measure*: it allows knowing how well the agent accomplishes its goals.
- *actuators*: the possible ways an agent can act in the system.
- *sensors*: the perceptions the agent collects from the environment.

In the following subchapters the classification of task environments is proposed. Then, according to this classification, a developed production environment is being classified. In the end of the chapter the brief description of the IAS process model “Lego-robots”, which is supposed to be used as an example of production environment, can be found.

2.1 Classification of Tasks Environments

According to [RuNo03], the tasks environments can be classified in the following dimensions:

- *Fully observable vs. partially observable.* If an agent's sensor gives it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable. Otherwise it is partially observable.

- *Deterministic vs. stochastic.* If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic.

- *Static vs. dynamic.* If the environment can change while an agent is deliberating, then we say the environment is dynamic.

- *Discrete vs. continuous.* The discrete/continuous distinction can be applied to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.

- *Single agent vs. multi-agent.* In multi-agent environments, agent-design problems arise as communication and stochastic behaviour. We can handle competitive or cooperative multi-agents environments.

2.2 Analysis of Developed Production Environment

The production process environments, coordination mechanism for which is being analyzed and developed in the context of this project, have following properties:

- *It is partially observable.* No agent within the system has total information about the whole environment.

- *It is stochastic.* The future state of the system cannot be predicted in the current moment, first of all because user may dynamically introduce new tasks

- *It is dynamic.* The environment can dynamically change during the work-time.

- *It is continuous.* The time is handled as continuous and it is impossible to define a finite number of states of the system.

- *It is multi-agent.* The system consists of multitude of different agents, interacting with each other.

2.3 IAS Process Model “Lego-robots” Description

The IAS process model was assembled and used in IAS as an example of production environment. It consists of the following parts:

- *Stations.* Process model object *station* models production plants. A special kind of station is *storage*, which models a storage that keeps workpieces.

- *Workpieces*. Process model object *workpiece* models 4workpieces that are being produced in the production environment. The workpieces should be transported between stations according to their own plans. The transportation is held by Lego-robots.

- *Lego-robots*. Process model object *Lego-robots* models a transporter that transports workpieces between stations.

More complete and concrete information on the IAS process model “Lego-robots” may be found in [Resc04].

3 Brief Analysis of the Agent-Oriented Methodology

In this chapter a brief analysis of the agent-oriented methodology is presented. First, the definition of agent is proposed. Then follow the concepts of agent-oriented methodology. Then goes the information on how agents can make the decisions.

The final point of the chapter is the analysis of how agents can be combined to multi-agent structures and interact with each other.

3.1 Definitions of Agents

There is no agreement in the literature what agent actually is. Instead, there are several definitions. As it is written in [RuNo03], there in the literature the following definition can be found:

- "An agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives" [Weis99]

- Software agents are the ones that have the following characteristics:
  - They are situated in some environment
  - They are capable of flexible autonomous action in order to meet design objectives. [JSW96]

- "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors." [RuNo03]

- "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act" [Maes95]

- "Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions." [Haye95]

- "...a hardware or (more usually) software-based computer system that enjoys the following properties:
  - Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
  - Social ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language;

  - Reactivity: agents perceive their environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the Internet, or perhaps all of these combined), and respond in a timely fashion to changes that occur in it;
  - Pro-activeness: agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by taking the initiative." [WoJe94]

- An agent is a delimitable software unit with defined goal. Being autonomous, an agent tries to achieve this goal and for that it interacts continuously with the environment and other agents [WGU03].

The last definition of an agent [WGU03] covers the most general aspects of agent's conception, so it was selected for this project. In this case the environment is a physical or software-technical field, in which agent is embedded and where agent acts.

3.2 Basic Concepts of Agent-Oriented Methodology

According to [WUG03] basic concepts of an agent-oriented methodology are (Figure 3.1):

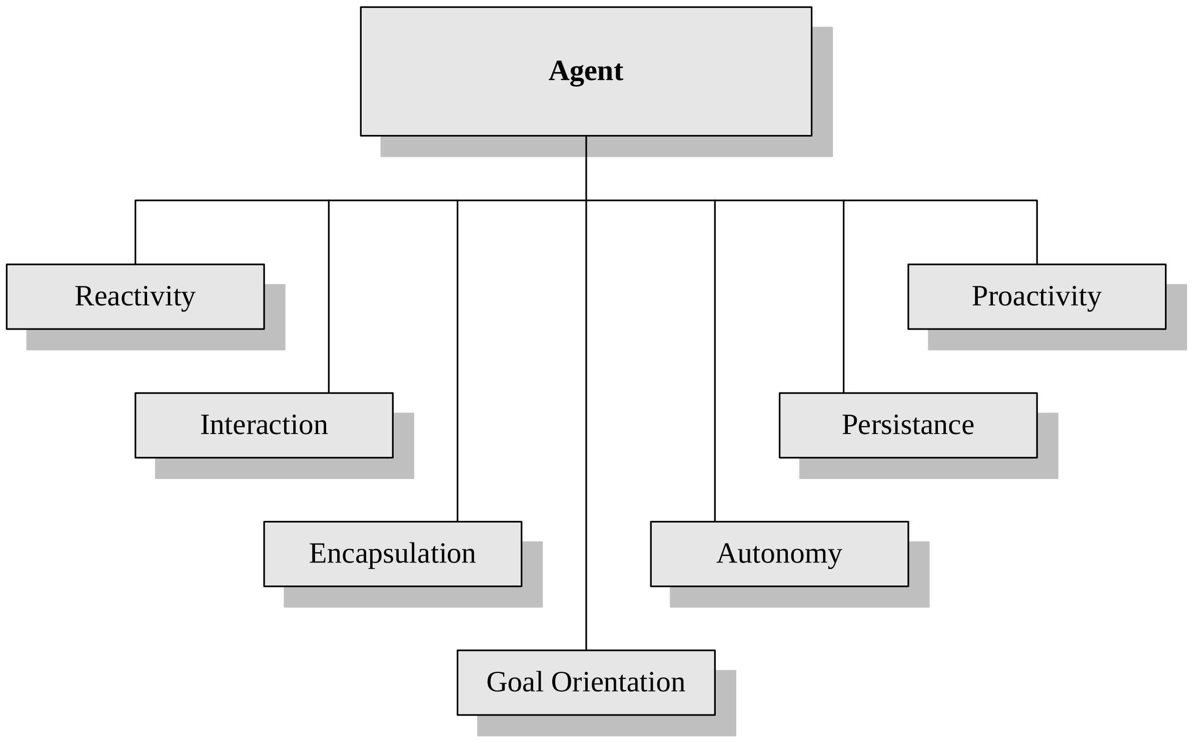

Figure 3.1: Basic Concepts of Agent-Oriented Methodology

- *Encapsulation:* state and behavior are combined in an encapsulated entity.

- *Goal orientation:* an agent orients his behavior to achieving defined goal. The goals can be defined either by developer or by user in run-time.

- *Reactivity:* reactivity is an ability of an agent to perceive it's environment and act in a suitable way.

- *Autonomy:* autonomy means the control over the internal state and the behavior. The behavior of agent is defined solely by agent and not from the outside an agent. Autonomy is also the precondition for proactivity.

- *Proactivity:* proactivity means the ability to act without direct influence from the outside. Precondition for proactivity is the existence of goals.

- *Interaction:* agents interact with each other to reach individual goals and to coordinate the use of common resources.

- *Persistence:* Agents have their own control flow. So, they are able to keep their internal state during the life cycle.

3.3 Decision-making Process

The agents need to have partial or total information of the world they interact with. The most effective way to handle observability of the world is for the agent to keep information about the part of the world it cannot see now. Agents also need the information how the world evolves independently of the agent and how the agent's own actions affect the world. In whole, this knowledge is called environment model.

The question is: how does agent make a decision what to do, according to its environment model? As it is written in [Fern03], depending on the way the agents make decisions how to act, agents may by classified in two most important groups:

- Goal-based agents:

The agents have goal information that describes the situations that are desirable. The agent program can combine this with information about the results of possible actions in order to choose actions that achieve the goal. Goal-based agents are flexible because the knowledge that supports its decisions is represented explicitly and can be modified (Figure 3.2)

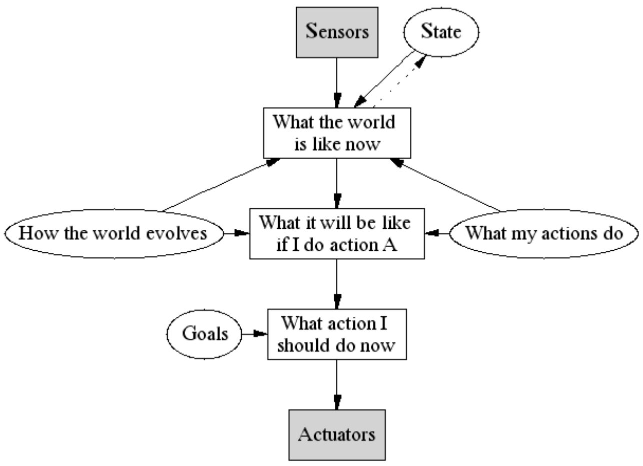
Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Goals
What action I should do now
Actuators

Figure 3.2: Goal-based agent

- Utility-based agents:

A utility function maps a state onto a real number, which describes the associated degree of success. The agent will try to maximize this degree of success. (Figure 3.3)

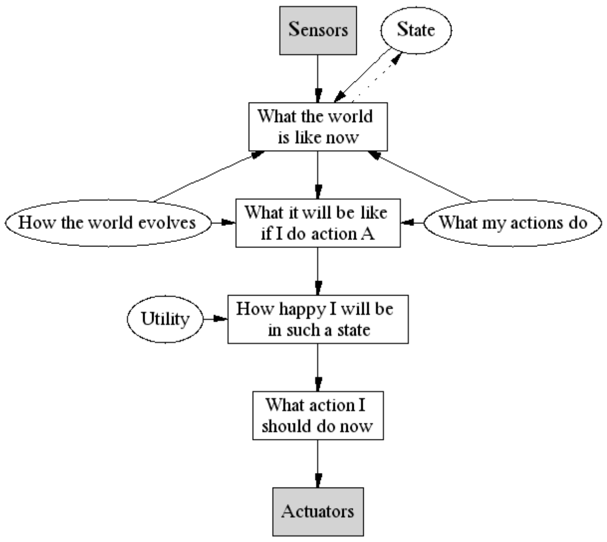
Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Actuators

Figure 3.3: Utility-based agent

Goal-oriented systems can show its best in relatively simple systems. In systems with significant resource sharing utility based approach is more relevant, because resource sharing requires a compromise, which can be easily transformed into numeric utility function.

3.4 Multi-Agent Systems

Agents are seldom stand-alone systems [Vlas03]. In many real-word problems they coexist and interact with other agents in different ways. Examples include software agents on the Internet, soccer playing robots, manufacturing processes control and many more. Such a system that consists of a group of agents that can potentially interact with each other is called a *multi-agent system* (MAS).

3.4.1 Characteristics of Multi-Agent Systems

The fundamental aspects that characterize MAS and distinguish it form a single-agent system are following [Vlas03]:

- *Agent design.* It is often that the different agents within the MAS are designed in different ways. In general, the design differences may involve the hardware (for example soccer robots based on different mechanical platforms), or the software (for example software agents running different operating systems). We often say that such agents are *heterogeneous* in contrast to *homogeneous* agents that are designed in an identical way and have a priori the same capabilities. However, this distinction is not clear-cut; agents that are based on the same hardware/software but implement different behaviors can also be called heterogeneous. Agent heterogeneity can affect all functional aspects of an agent from perception to decision making, while in single-agent systems the issue is simply nonexistent.

- *Environment.* Agents have to deal with environments that can be either *static* (time-invariant) or *dynamic* (nonstationary). In MAS the presence of multiple agents makes the environment appear dynamic from the point of view of each agent. This can often be problematic, for instance in the case of concurrently learning agents where non-stable behavior can be observed. There is also the issue which parts of a dynamic environment an agent should treat as other agents and which not.

- *Perception.* The collective information that reaches the sensors of the agents in a MAS is typically distributed. The agents may observe data that appear at different locations, arrive at different times, or even require different interpretations. This automatically makes the world state partially observable to each agent, which has various consequences in the decision making process of the agents. An additional issue is sensor coordination, that is, how the agents can optimally combine their perceptions in order to increase their collective knowledge about the current state.

- *Control.* Contrary to single-agent systems, the control in MAS is typically distributed (decentralized). This means that there is no central process that collects information from each agent and then decides what action each agent should take. The decision making of each agent lies to a large extent within the agent itself. The general problem of multi-agent decision making is the subject of coordination, which will be described in the following chapters. In general, coordination ensures that the individual decisions of the agents result in good joint decisions for the group.

- *Knowledge.* In single-agent systems it is typically assumed that the agent knows its own actions but not necessarily how the world is affected by its actions. In MAS, the levels of knowledge of each agent about the current world state can differ substantially. For example, in a team MAS involving two homogeneous agents, each agent may know the available action set of the other agent, both agents may know (by communication) their current perceptions, or they can infer the intentions of each other based on some shared prior knowledge. On the other hand, an agent that observes an adversarial team of agents will typically be unaware of their action sets and their current perceptions, and might also be unable to infer their plans. In general, in MAS each agent must also consider the knowledge of each other agent in its decision making. A crucial concept here is that of common knowledge, according to which every agent knows a fact, every agent knows that every other agent knows this fact, and so on.

- *Communication.* Interaction is often associated with some form of communication. Typically the communication in MAS is as a two-way process, where all agents can potentially be senders and receivers of messages. Communication additionally raises the issues of what network protocols to use in order for the exchanged information to arrive safely and timely, and what language the agents must speak in order to understand each other (especially if they are heterogeneous).

3.4.2 The reasons to use Multi-Agent Systems

As it is written in [GHN+97], the motivations for the use of MAS are caused with their ability:

- to solve problems that are too large for a centralized single agent to do due to resource limitations or the sheer risk of having one centralized system;

- to allow for the interconnecting and interoperation of multiple existing legacy systems, e.g., expert systems, decision support systems, etc.;
- to provide solutions to inherently distributed problems;
- to provide solutions where the knowledge basis is distributed;
- to enhance speed (if communication is kept minimal), reliability (capability to recover from the failure of individual components, with graceful degradation in performance), extensibility (capability to alter the number of processors applied to a problem), the ability to tolerate uncertain data and knowledge;
- to offer conceptual clarity and simplicity of design.

3.4.3 Multi-Agent Systems classification depending on the degree of cooperation

Depending on the degree of cooperation exhibited by the individual agents, two types of multi-agent systems may be distinguished [GHN+97].

- In *cooperative multi-agent-systems* agents are supposed to bring the benefit to the global problem resolution. Usually these agents are programmed by the same designer, who was only concerned with increasing the general system's performance and not the performance of individual agents. Hence such agents are considered cooperative and the whole problem-solving process is called cooperative distributed problem-solving process.
- *Competitive or self-interested multi-agent systems* are concerned with individually motivated agents who probably had been designed by independent designers. When considering system behavior, it is not possible count on agents cooperating just because they would be designed that way. What was important for agents is the benefit they could derive from their actions. So, the purpose of all individual agents is to maximize their own benefit. Hence such agents are considered self-interested, competitive or non-cooperative.

3.4.4 Challenging issues

The transition from single-agent to multi-agent systems has significant advantages, but also raises a lot of problems to resolve. Due to [Vlas03] there are following issues:

- How to decompose the problem, allocate subtasks to agents, and synchronize the results;
- How to handle the distributed perceptual information and how to enable agents to maintain shared models of the world;
- How to implement decentralized control and build efficient coordination mechanisms among agents;
- How to design efficient multi-agent planning and learning algorithms;
- How to represent knowledge and enable agents to reason about the actions, plans and knowledge of other agents;

- How to realize communication process, which communication languages and protocols to use, what, when and with whom should an agent communicate;

- How to enable agents to negotiate and resolve conflicts;

- How to enable agents to form organizational structures like teams or coalitions, how to assign roles to agents;

- How to ensure coherent and stable system behavior.

Those problems presented above are not independent, and their solutions can affect each other. For example, a distributed planning activity requires a particular coordination mechanism, and so on.

3.5 Gaia Methodology Description

Theory of multi-agent systems defines which structure multi-agent system may have and which benefit such structure will bring. However it does not define *how* to develop such structure. With this leads different iterative approaches towards modeling and developing agent-based systems. Among them are Gaia Methodology (Gaia), The Multi-Agent Systems Engineering methodology (MaSE), etc.

The Gaia methodology is presented in [WJK00]. According to the description, proposed in [Tvei01], for analysis and design in this project a Gaia methodology will be used. Gaia is a general methodology that supports both the micro-level (agent structure) and macro-level (agent society and organization structure) of agent development. Gaia approach requires that inter-agent relationships and agent abilities are static at run-time. The motivation behind Gaia is that existing methodologies fail to represent the autonomous and problem-solving nature of agents; they also fail to model agents' ways of performing interactions and creating organizations. Using Gaia, software designers can systematically develop an implementation-ready design based on system requirements.

The first step in the Gaia is find the *roles* in the system. The second one is to model *interactions* between the roles found. Roles consist of four attributes: *responsibilities*, *permissions*, *activities* and *protocols*. Responsibilities are of two types: *liveness properties* - the role has to add something good to the system and safety properties – prevent and disallow that something that something bad happens to the system. *Permissions* represents that the role is allowed to do, for example, which information it is allowed to access. *Activities* are tasks that a role performs without interacting with other roles. *Protocols* are the specific patterns of interaction, e.g. a seller role can support different auction protocols, e.g. "English auction". Gaia has formal operators and templates for representing roles and their belonging attributes; it also has schemas that can be used for the representation of interaction.

In the Gaia *design* process, the first step is to map roles into *agent* types, and then to create the right number *agent instances* of each type. The second step is to determine the *services model* needed to fulfill a role in one or several agents, and the final step to is create the *acquaintance* model for the representation of communication between the agents.

According to [Fern03] Gaia approach can be graphically represented like it is done in Figure 3.4.

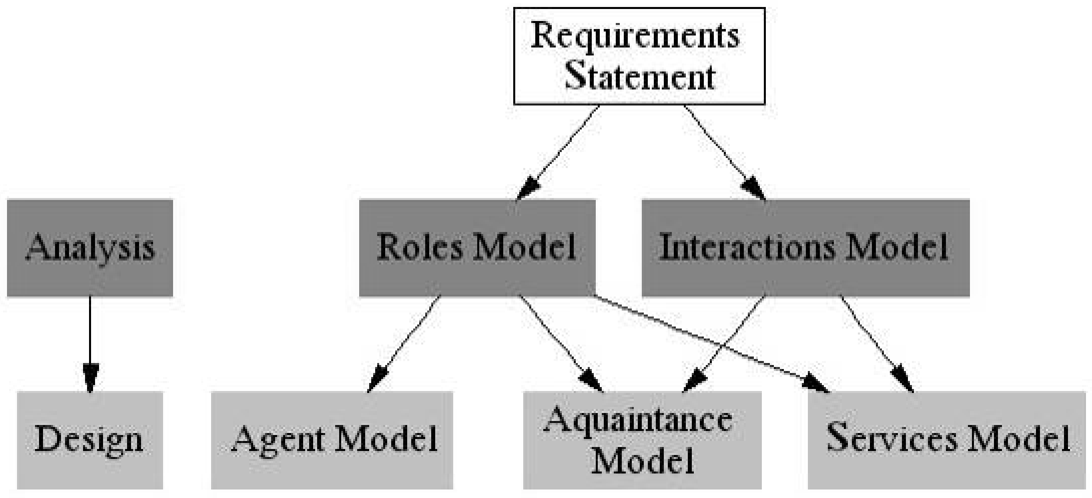

Figure 3.4: Gaia methodology diagram

Due to mentioned restrictions of Gaia of less value in the open and unpredictable environments, on the other hand it has been proven as a good and clear approach for developing closed domain agent-systems. That is why it was selected for agent-oriented analysis and design in this project.

4 Analysis of Coordination Techniques

The problem of coordination is a central point in multi-agent systems. Without it a group of interacting agents turn’s into a collection of individuals, that are not able to fulfill a global goal. Coordination has been studied by researches in different brunches, such like social science, political science, social psychology, sociology, etc.

So, what is actually coordination? According to [GHN+97] *coordination* is a process by which agents engage in order to ensure their community acts in a coherent and harmonious manner. In other words coordination make multi-agent system act like one unit.

In the following subsections there are the motivations to use coordination, formulation of the properties of coordination subsystem of multi-agent system’s software and the classification of in the literature known coordination techniques.

4.1 Reasons for coordination

In Accordance with [NLJ96] there are several reasons why multiple agents need coordination:

- *To prevent chaos*

Coordination is necessary because, with the decentralisation in agent-based systems, chaos can set in easily. As a result no unit has a global view of the entire environment any more. Agents only have local views, goals and knowledge which may conflict with others. That is why a group of agents need to be coordinated.

- *To meet global constraints*

Usually there are global constraints which a group of agents must satisfy to be successful. Agents need to coordinate their behaviour if they are to meet such global constraints.

- *To distribute an experience, resources or information*

Agents may have different capabilities and specialised knowledge on the same subject. Also, they may have different sources of information, resources, reliability levels, responsibilities, limitations, charges for services, etc. In such scenarios, agents have to be coordinated to exchange necessary information.

- *To resolve a dependencies between agent's actions*

Agent's goals are frequently interdependent. Consider two agents solving the trivial blocks problem shown in Figure 4.1.

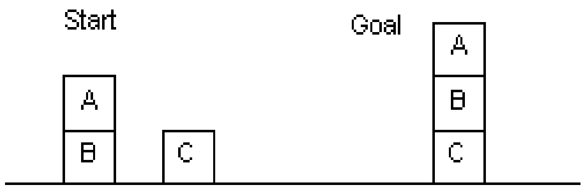

Figure 4.1: A Blocks Problem

The easiest way to solve it would be for the first agent to take on the sub-goal of stacking B on C while the second stacks A on top of the stack B-C in order to achieve A-B-C. Clearly, the sub-goals are interdependent: the second agent has to wait for the first agent to complete its sub-goal before it can perform its own. Where such interdependencies exist the activities of the agents must be coordinated.

- *Efficiency*

Even when individuals can function independently, thereby obviating the need for coordination, information discovered by one agent can be of sufficient use to another agent that both agents can solve the problem twice as fast.

Coordination, in turn, may require cooperation. However the cooperation among a set of agents does not necessarily lead to coordination. Indeed, it may even result in incoherent behaviour. This is because for agents to cooperate successfully, typically, they must maintain models of each other as well as develop and maintain models of their future interactions. If agent's beliefs about each other are wrong, that may lead to incoherent behaviour.

Coordination may also occur without cooperation. Likewise, non-cooperation among agents does not necessarily lead to incoherent behaviour.

To achieve coordination, agents may have to *communicate* with one another. However, agents may achieve coordination without communication, using the models of each other's behaviours. In such a situation, coordination can be achieved via organisation.

Thus, with or without cooperation, with or without communication, agents within one multi-agent system should know each other's goals, intentions, results and state, so they need to be coordinated.

4.2 Desired Properties for Cooperative Problem-Solving

While developing the multi-agent collaborative system, which uses coordination and communication for its activity, the principles of the coordination should be first analyzed and defined. Due to [WoJe99] this theory should fulfill following requirements:

- Agents are autonomous

Agents are Autonomous Problem Solvers. Hence they will take part in cooperative activities only if they *choose* to do so. A theory that simply required agents to cooperate whenever they were asked would not be able to model the coordination process which occurs in a real world in an adequate way.

- Cooperation can fail

A corollary of the fact that agents are autonomous is that cooperation may fail. If agents must cooperate on any demand, it means that sometimes they won't. Even when initial cooperation is established, it can fail for many different reasons. For example, a group of agents that agree to cooperate in principle may discover that the assumptions upon which their decision to cooperate was made is not correct. Alternatively, another activity of agent beyond the control of the team may make taking part in cooperation within the team worthless or even impossible. An adequate theory of coordination must recognize that such situation is possible, identify the key points at which it may occur, and characterize the behavior of agents in such circumstances.

- Communication between agents is essential

Although something cooperation is possible without communication, communication still remains so fundamental for the process of cooperation that coordination theory should describe when and where communication should take place. So, it should *predict* communication.

- Agents initiate coordination processes

Cooperation does not come from a vacuum. It occurs because a group of agents consider they will in some way benefit from it. That is why an adequate theory of coordination should take into account the circumstances under which agents will begin to initiate collaboration processes.

- Agents are supportive by their nature

Cooperating agents have to support each another during the fulfillment of the global task. That means that agents execute their part of the team's action, and do typically what they can to ensure that the remainder of the team does likewise. The theory of coordination must describe the type's support, when it should occur, and what form such support should take.

- Agents are reactive

Any more o less realistic environment is dynamic. Agents must take this into account, and respond to any changes that affect their plans. An adequate coordination theory must therefore recognize the reactive aspect of agent's behavior, and characterize the behavior of the agents in such circumstances.

4.3 Coordination Techniques Classification

There can be different approaches which have been devised to achieve coordination in multi-agent systems. Different approaches may have different coordination intensity.

As stated in [GHN+97], there are 4 approaches to solving coordination problem in multi-agent systems:

- *Organisational Coordination*
- *Contracting for Coordination*
- *Multi-Agent Planning for Coordination*
- *Social Laws for Coordination*

[NLJ96] classifies coordination techniques in the similar and defines following 4 categories:

- *Organisational Structuring*
- *Contracting*
- *Multi-Agent Planning*
- *Negotiation*

The summarizing, based on these two classifications, is provided below (Figure 4.2).

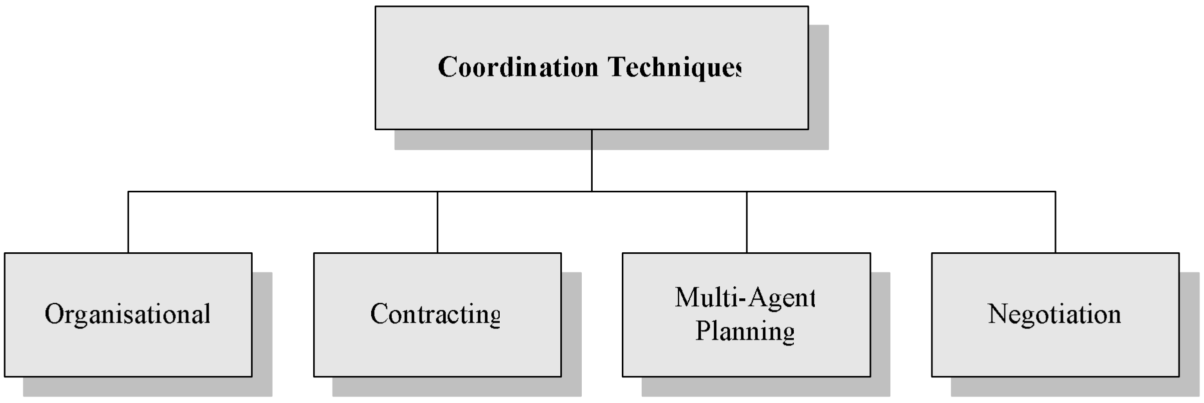

Figure 4.2: Coordination Techniques

4.3.1 Organisational Coordination

4.3.1.1 Description

It is the simplest coordination approach which suppose the *a priori* organisational structure. On the other words, organisation defines implicitly the agent's responsibilities, capabilities, connectivity and control flow. It also defines roles, communication paths and authority relationships. Thus, with organisational coordination long-term relationships between agents are predefined.

Organisational coordination is typically based on *client-server* architecture and used for task and resource allocation among slave agents by some master agent. This technique may be implemented in a couple of ways:

- The master agent plans and distributes fragments of the plan to the slaves. The slaves may or may not communicate amongst themselves but must ultimately report their results to the master agent. So, while the master has full autonomy with respect to the slaves, the slaves have only partial autonomy with respect to their master.

- The second way is blackboard coordination which uses the classic blackboard architecture to provide a coordinating base. In this scheme agents post to and read from the general blackboard. The scheduling agent (or master agent) schedules the agents' read/write to/from the blackboard. This approach may be used when the problem is distributed and a central scheduling agent is present or when tasks have already been assigned, *a priori*, to agents.

It is worth to say that organisational coordination usually should not be always associated with hierarchies, because coordination may also occur amongst agents.

4.3.1.2 Analysis

Organisational Coordination is rather simple by it's nature, so it can be simply implemented in systems without intensive dynamic exchange of information. It also, in general, doesn't lead to significant communication load.

The disadvantage of organisational coordination is the fact that much control is done by the slaves' actions. However, such control contraries to all the benefits of distributed problem-solving process: speed (because of parallelism), reliability, concurrency, robustness, minimal bottlenecks, etc. In the blackboard coordination scheme, with no direct agent-to-agent communication, a severe bottleneck may result if there are too many agents. Also, such centralised control as in the master-slave technique is contrary to the basic assumptions of distributed problem-solving process. It presumes that at least one agent has a global view of the entire agency - in many domains, this is an unrealistic assumption. Thus, distributing trivial or small tasks can be more expensive than performing them in one location.

Thus, the organisational coordination is most useful in:
- multi-agent systems where master-slave relationships are being modelled;
- in systems where a priori defined relations between agents exists.

4.3.2 Contracting Coordination

4.3.2.1 Description

A classic coordination technique for task and resource allocation among agents and determining organisational structure is the *contract net* approach. It usually uses or is based on *Contract-Net Protocol*.

The basis of this approach is that if an agent cannot solve the problem itself, it will decompose the problem into subproblems and try to find the agents with necessary resources to solve these subproblems. So, it is assumed that agents can play two roles:

- A manager who breaks a problem into sub-problems and searches for contractors to do them and monitor the problem's solution process;

- A contractor who does a sub-task.

The contractors also may recursively become managers and further decompose the subtask and sub-contract them out to other agents.

Managers locate contractors via a process of bidding which proceeds as follows:
1. Manager announces a task;
2. Contractors evaluate the task with respect to their abilities and considerations;
3. Contractors send bids to the manager;
4. Manager evaluates received bids, chooses a contractor and awards the contract to it;

5. Manager waits for the result of the contract.

4.3.2.2 Analysis

Contracting coordination is a completely distributed scheme where a node can both simultaneously be manager and contractor.

The main advantage of the contracting coordination is hat it uses dynamical task allocation via self-bidding. That leads to better arguments, agents can be added and removed dynamically, it also provides natural load-balancing.

The disadvantages involve the fact that contracting coordination does not detect or resolve conflicts, the agents in the contract net are supposed to be rather passive and benevolent, which is not typical for real-world scenarios, and finally such type of coordination makes a rather significant communication load.

Thus, contracting coordination is best used when:
- the task has a well-defined hierarchical nature;
- it is possible to make at least a coarse-grained decomposition of the task;
- there is minimal coupling among subtasks.

4.3.3 Multi-Agent Planning

4.3.3.1 Description

By multi-planning approach the agents are engaged in multi-agent planning. In order to avoid inconsistent or conflicting actions and interactions and organize collaboration, agents build a multi-agent plan that details all their future actions and interactions required to achieve their goals, and carry out the execution of plans with more consequent planning and re-planning.

In general, there are two types of multi-agent planning:

- Centralized multi-agent planning
- Distributed multi-agent planning

In centralized multi-agent planning, the separate agents form their individual plans and then send these plans to a central coordinator, who analyses them in order to identify potential inconsistencies and conflicting interactions. The idea behind this approach is that the central coordinator can identify critical regions of plans around which agents should synchronize and insert plan steps for sending and waiting for synchronization messages to ensure proper synchronization. After that the individual partial plans can be merged into a multi-agent plans.

The distributed technique for multi-agent planning foregoes the use of a central coordinator, and instead provides each agent with a model of other agents' plans. Agents communicate in order to build and update their individual plans and their models of others' until all conflicts are removed.

4.3.3.2 Analysis

The advantages of the multi-planning is the possibility to implement centralized or decentralized system with strong planning capabilities. Forming multi-agent plans allows to determine all the actions and interactions beforehand.

As a disadvantage, multi-agent planning of whatever form requires that agents share and process substantial amounts of information; hence, it is likely to require more computing and communication resources comparing with other approaches. The centralized multi-agent planning technique also shares many of the limitations of the master-slave coordination technique. The second point is that the coordination in distributed multi-agent planning is much more complex than in the centralized form as there may not be any agent who possesses a global view of the distributed system.

Multi-planning approach can prove itself on the best way on the multi-agent systems, that:

- have a prior structure, that does not changes intensively;
- need a advance planning of subtasks execution

4.3.4 Negotiation

4.3.4.1 Description

A significant part of researches about the coordination is to understand and model the way how humans coordinate their activities through negotiation. The common idea in all DAI contributions to negotiation is that agents use negotiation for coordination. Though negotiation is very important for the modeling of multi-agent systems

Actually, there is no clear and common definition of what negotiation is. One of the definitions is that by [BuMu92]: *Negotiation is the communication process of a group of agents in order to reach a mutually accepted agreement on some matter.*

Agreement might be about anything agents deal with: meeting place or time, price, a joint action, or a joint objective. The search process may involve the exchange of information, the relaxation of initial goals, concessions, etc. Hence, the basic idea behind negotiation is reaching a consensus.

It is worthwhile to mention that difference between the negotiation and the other coordination approaches is quite indistinct. Indeed, the coordination approaches, described above, cover most of the coordination activity that can be called negotiation. The main difference consists not in *what* agents do to be coordinated but *how* agents do that. By negotiation agents must reason about beliefs, desires and intentions of other agents [RaGe95]. This has lead to the usage of all sorts of AI and mathematical techniques including logic, case-based reasoning, belief revisions, optimisation and game theories, etc.

There may be several kinds of negotiation:

- Game theory-based negotiation;
- Plan-based negotiation;
- Human-inspired and miscellaneous AI-based negotiation approaches.

The key concepts in this game theory approach to negotiation are following: utility functions, a space of deals, strategies and negotiation protocols. *Utility* is defined as the difference between the worth of achieving a goal and the price paid in achieving it. A *deal* is an action an agent can take which has an attached utility. The negotiation *protocol* defines the rules which govern the negotiation, including how and when it ends (e.g. by agreement or no deal). The actual negotiation proceeds as follows. Utility values for each outcome of some interaction for each agent are built into a payoff matrix, which is common knowledge to both parties involved in the negotiation. The negotiation process involves an interactive process of offers and counter-offers in which each agent chooses a deal which maximises its expected utility value. Thus, as a result of the negotiation process an optimal strategy of actions is selected.

Plan-based negotiation is proposed in [KrMa91]. Negotiation is carried out as a two stage process: first, agents plan their activities separately, and then secondly, they coordinate their plans. The coordination of all the agents' plans is done by a separate coordination agent, though they note that this role may be played by any of the agents. The negotiation protocol is presented in terms of agent's states, message types and conversation rules between agents.

It appears that almost every form of human interaction requires some degree of explicit or implicit negotiation. Hence, it is not very surprising that many negotiation researchers draw from human negotiation strategies. As noted earlier, these often lead to the usage of miscellaneous AI techniques including logic, case-based reasoning (CBR), constraint-directed search, etc. Some of the examples are provided below.

In [BuMu92] a negotiation approach proposed, which draws from socio-psychological theories of negotiation. It evolves a cyclic negotiation model which is both general and simple. The cyclic nature of the model addresses the thorny issue of conflict resolution. The general strategy is that negotiation begins with one, some or every agent making a proposal. Then agents evaluate and check the proposals against their preferences, and criticise them by listing their preferences which are violated by the proposals. The agents then update their knowledge about other agents' preferences and the negotiation cycle resumes with a new proposal or proposals in the light of the newly gleaned information. Conflicts between agents are handled in a concurrent conflict resolution cycle.

According to [SaFo89] the negotiation may be viewed as a *constraint-directed search* of a problem space using negotiation operators. Initially, preferences of negotiation are modeled as constraints. Negotiation is composed of two phases: a communication phase, where all information is communicated to participating agents, and a bargaining phase where deals are made between individuals or in a group. In their approach, agents negotiate via relaxing various conflicts and constraints until agreement is reached. Alternatively, the solutions may be modified. The negotiation operators are drawn from human negotiation studies and include operators which simulate relaxation, reconfiguration and composition, which are used to generate new constraints.

A knowledge-based model of an incremental form of negotiation is proposed in [Werk90]. It is based largely on various human models of negotiation. This scheme uses a shared knowledge representation called *shareable agent perspectives* which allows agents to perform negotiation in a manner similar to cooperating (or competing) experts who share a common background of domain knowledge. Essentially, it exploits a blackboard with partitions for requested proposals, rejected proposals, accepted proposals, a communications partition and shared knowledge. Such rich detail and knowledge of the perspectives of other agents provide valuable information for agents to make better proposals in the future. Negotiation is done in a three-phase cycle. The first phase involves some proposing agent announcing a proposal which is received and evaluated by the receiving agent. The second phase involves generating a counter proposal if the latter is not happy with the initial proposal, or it may be simply accepted. The third phase involves the submission of the counter proposal for review by other agents.

4.3.4.2 Analysis

Game theory-based negotiation is supposed to give a correct and proper model of a real word collaboration process, based on negotiation. Indeed, it gives a pared-down model of how human being made a decision-making process. On the other hand game theory-based negotiation fails to address some crucial issues. Firstly, agents are presumed to be fully rational and acting as utility maximisers using pre-defined strategies. Secondly, all agents should have knowledge of the payoff matrix, and therefore should have full knowledge of the other agent’s preferences - this is certainly unlike the real world where agents only have partial or incomplete knowledge of other agent’s beliefs. Furthermore, the payoff matrix could become very large and intractable for a negotiation involving many agents and outcomes. Thirdly, agents only consider the current state when deciding on their deal: past interactions and future implications are simply ignored. Hence, despite the mathematical proof of efficiency of game-theory based negotiation, provided in [RoZl94], it is unlikely that game theory-based negotiation will suffice for real-life, industrial agent-based applications for the reasons already offered. In brief, its assumptions are untenable in real applications.

The plan-based negotiation is similar to multi-planning coordination, however in contrast to multi-planning coordination the plan-based negotiation does not define so strict how coordination process will be carried out. It gives more freedom to the developers, but on the other hand, as it is pointed out in [BuMu92], it does not really present a negotiation model but just prescribes one. It is left to the agents how they will achieve consensus. In general, plan-based negotiation suffers from the limitations of centralised or distributed multi-agent planning, depending of which of this two types is used.

Human-inspired and AI-based approaches are quick upcoming ones, and are suspected to be a basis for future coordination techniques development. Indeed, they use the most realistic and adequate model of the real word and the distributed problem-solving process is the closest to the one that occurs in real life. However, in the present-day state human-inspired and AI-based approaches are rather abstract and theoretical, which leads to a significant difficulties in implementing coordination techniques based on these approaches in manufacturing problems.

5 Approaches to Coordination in the Production Environment

The classification of the coordination approaches presented in previous chapter is rather general. The real approaches to the coordination for the production environment can joint the properties of different approaches. So, it is worthwhile to describe, how coordination techniques can be applied to given production environment.

The goal of the coordination within the system is to find such mapping between workpiece transportation tasks and the robots that transport them, that benefit the highest utility (Figure 5.1)

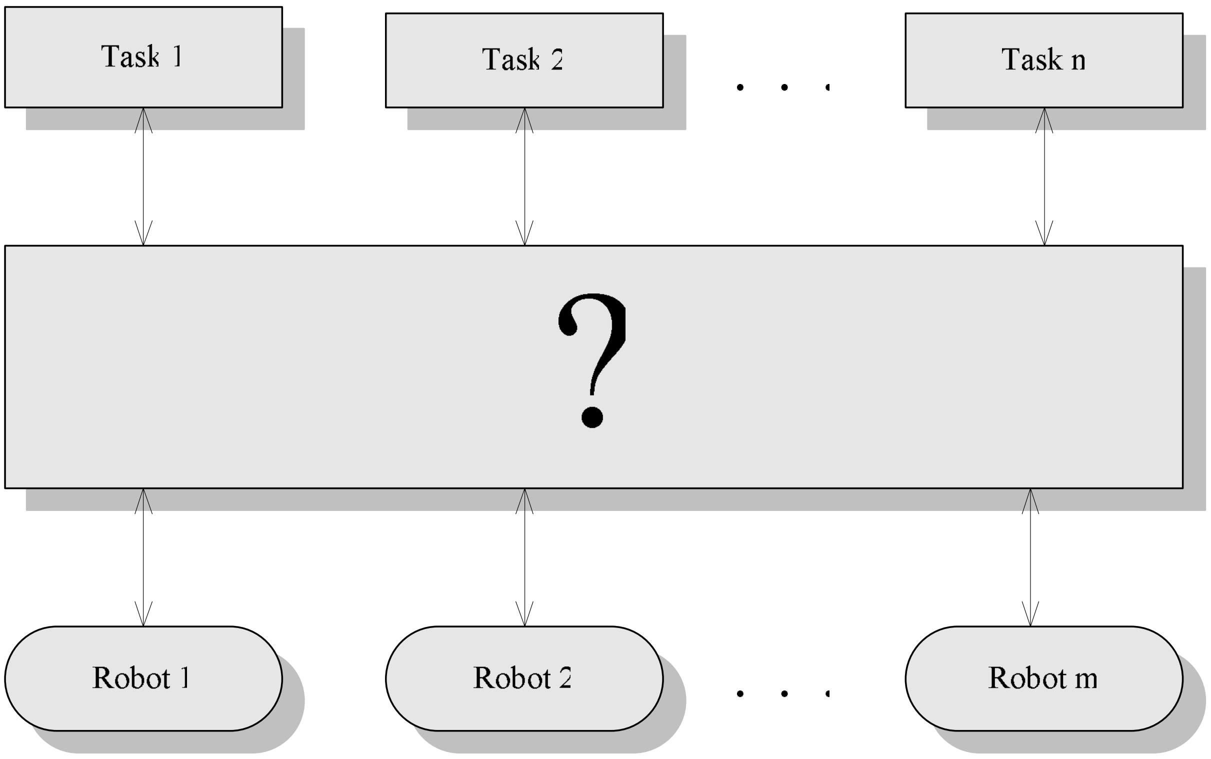

Figure 5.1: Mapping between transportation tasks and robots

Below follows 4 key concept for coordination that can be implemented in multi-agent control system for production environment (Figure 5.2).

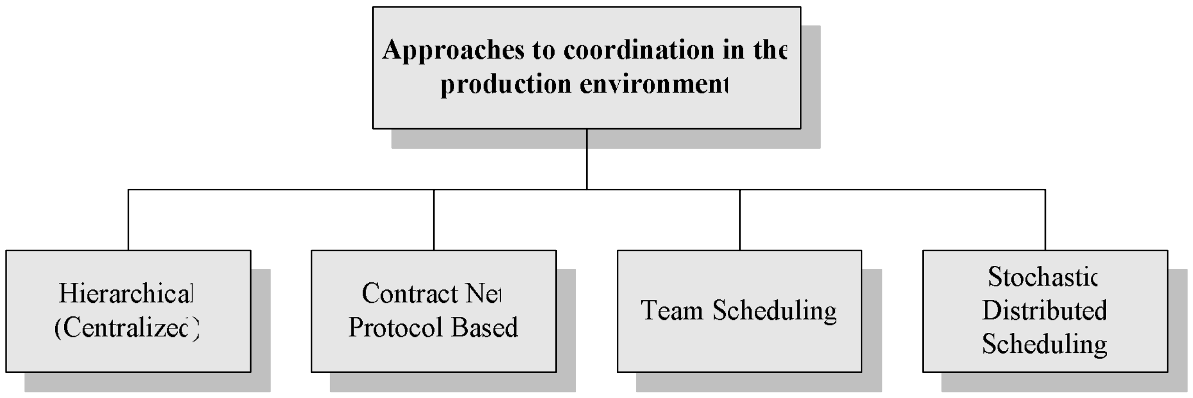
Approaches to coordination in the production environment
Hierarchical (Centralized)
Contract Net Protocol Based
Team Scheduling
Stochastic Distributed Scheduling

Figure 5.2: Approaches to coordination in the production environment

5.1 Hierarchical (Centralized) Approach

5.1.1 Description

Hierarchical or centralized approaches are based on organizational coordination approach and respects from multi-agent system to have a central coordinating agent (Figure 5.3)

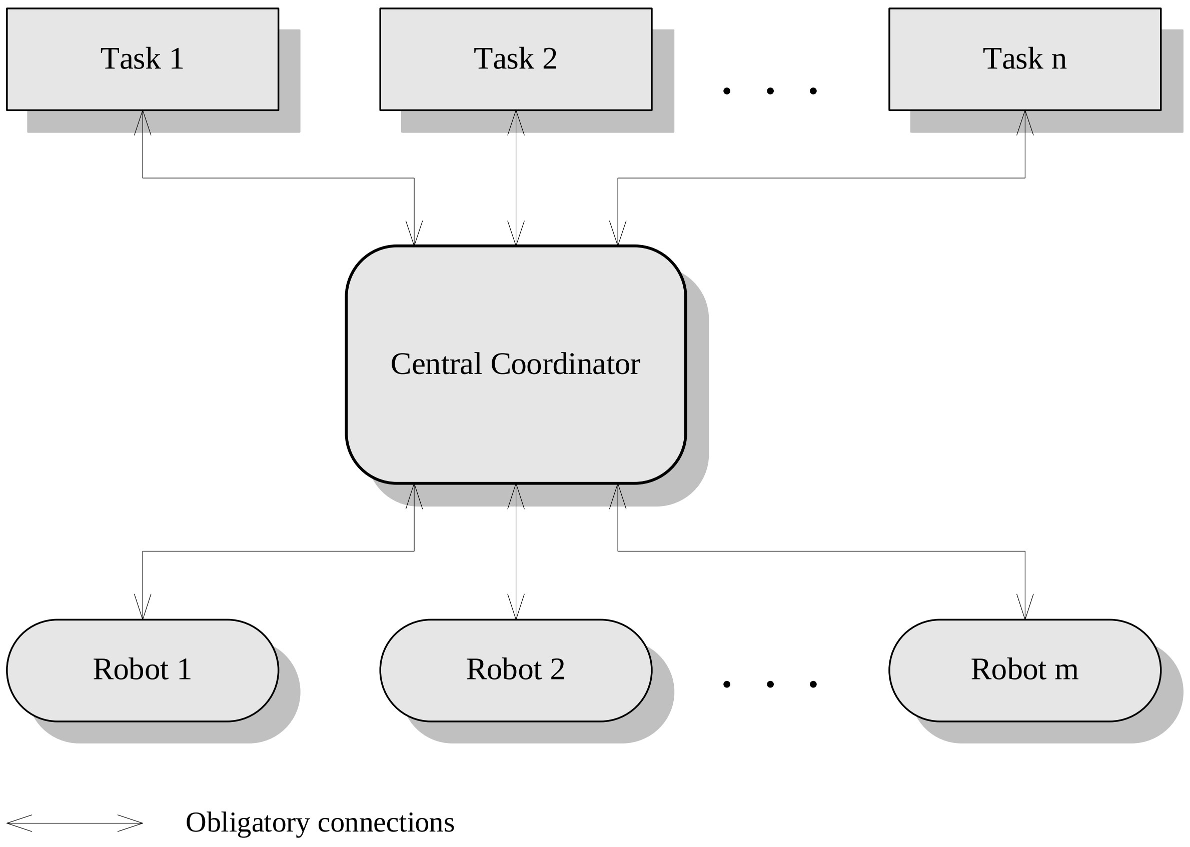

Figure 5.3: Hierarchical (Centralized) Approach

The goal of this central coordinator is to make and then keep up to date the schedule of all transportation activities. To do that, it has to:

- first receive information from a workpiece about the tasks that should be carried out;
- after receiving the new information to make/remake a mapping of workpiece transportation tasks and robots.

The above described coordination approach was implemented in [Resc04].

5.1.2 Advantages

The approach to coordination in production multi-agent systems inherits all the advantages and disadvantages from the organizational coordination.

In particular such coordination approach seems to be the simplest one. So, it is relatively easy to implement hierarchical approach in a real prototype.

It also leads to a moderate communication load. Indeed, the workpieces contact the coordinator agent only once to announce the transportation task. Robots are also contacted only once, when the global schedule is ready.

The last point is the fact that hierarchical organization guarantees the most accurate schedules, comparing with the other approaches. Indeed, the coordinator has all the knowledge about the environment, so he has everything to do the optimal schedule.

5.1.3 Disadvantages

The first point is the low reliability. With the failure of coordinator agent the whole system becomes inoperative. That is typical disadvantage for all centralized systems.

Secondly, although the global coordination load is comparatively low, all the communications go through the coordinator object. Thus, with an increase of amount of agents in the system, coordinator may become a bottleneck of the whole system. It, in its turn leads to decreasing of speed and reliability.

At last, such coordination assumes that coordinator agent has a global view of the whole environment. That is completely impossible in big domains.

5.2 Contract Net Protocol Based Approach

5.2.1 Description

Contract net protocol based approaches to negotiating are based on contracting coordination approach.

By contract based approaches it is supposed that workpiece transportation task is assigned to a random robot. Then, the robot tries to find another robot that can do this transportation task better then him, or, in other words, who can bring a higher utility executing this task. In case when such one is found and this one is agree to take the task, then the task is passed to this agent. The graphical presentation of an approach may be found on Figure 5.4.

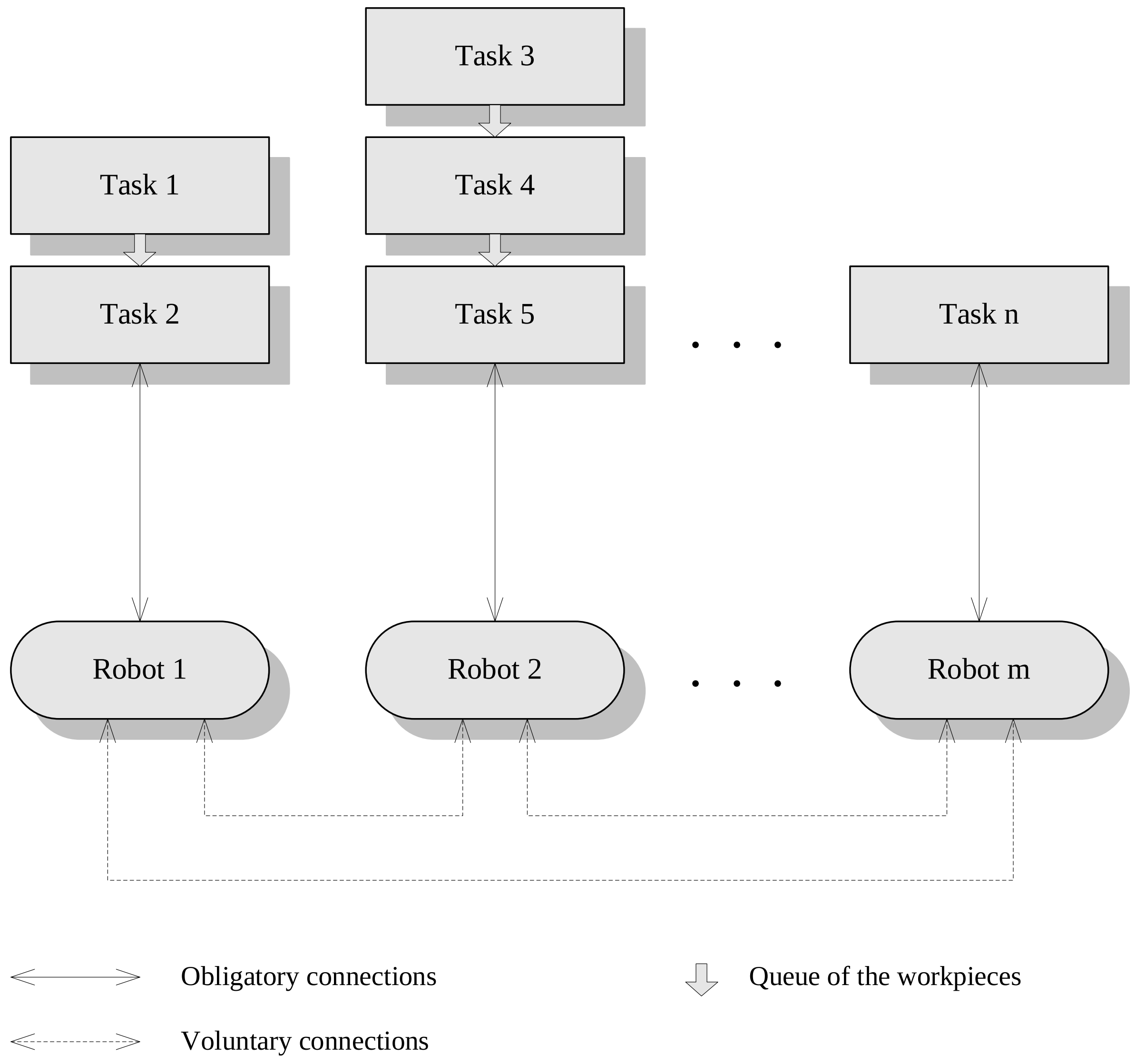

Figure 5.4: Contract Net Protocol Based Approach

The contract net protocol based approach was used, for example, in [MuBo96] in a multi-agent system for distributed scheduling for distributed processing.

5.2.2 Advantages

The contract net protocol based approach is decentralized by its nature, thus it the system implemented with the contract net based coordination approaches will be more reliable and quick-working.

Contract net protocol also allows a completely dynamical task allocation. It makes possible to add and remove agents in run-time.

Contract net protocol based coordination also provides natural load-balancing capabilities.

5.2.3 Disadvantages

However the total decentralization is also a disadvantage, because it also makes very difficult to resolve conflict between robots, for example, in case of resource allocation.

The second point is the significant communication load, which occurs because of a numerous interactions between robots.

5.3 Team Scheduling Approaches

5.3.1 Description

Team scheduling approaches are based on multi-planning coordination approaches with elements of negotiation activities.

In general, the approach works in the following steps:
1. Somehow the workpiece transportation tasks are distributed among the robots. The more close the initial distribution is to optimal distribution, the better.
2. After the initial distribution is built up, the optimization of plans begins. During the optimization robots interact with each other or with workpieces and iteratively change the mapping between the workpiece transportation tasks and robots.

The graphical presentation on how team scheduling approaches work is shown below (Figure 5.5).

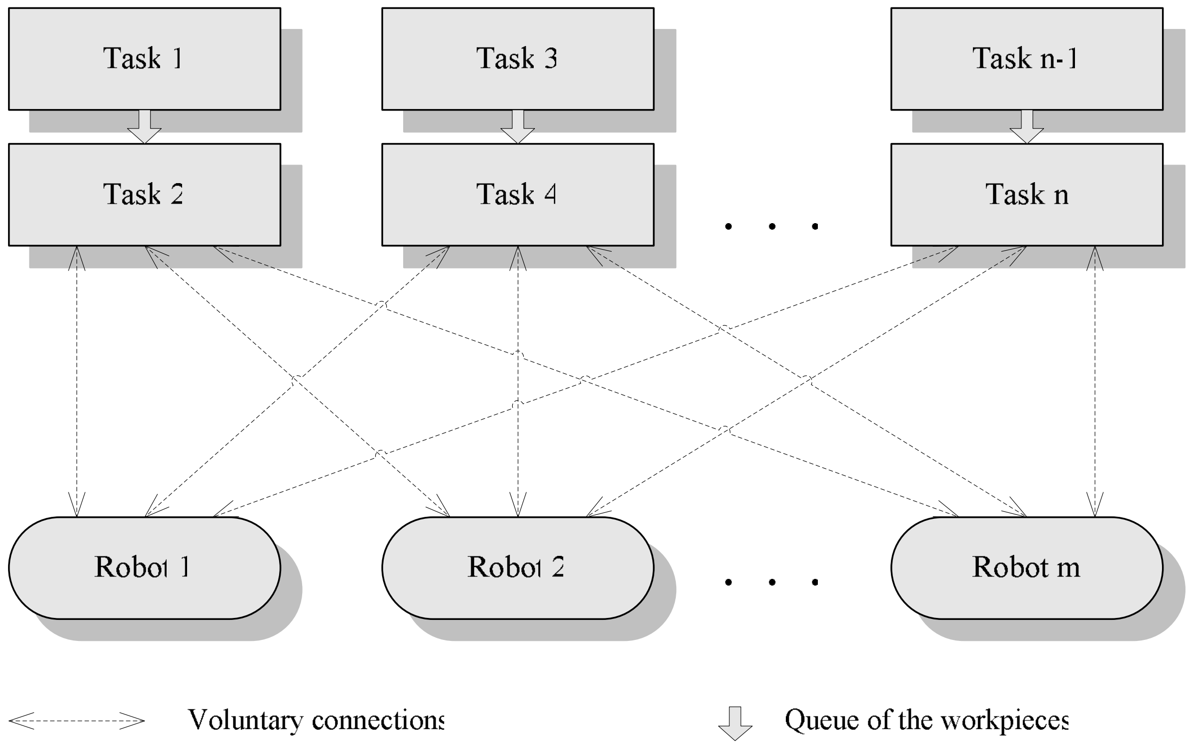

Figure 5.5: Team Scheduling Approach

There are 2 very important questions for team scheduling approaches:

- *How* to make the initial distribution, a first approximation, so that it would be as close as possible to the optimal one?
- *How* to carry out mentioned above optimization?

There are no general answers to these questions suitable for all types of the environments. Some ideas on how to make a first approximation and do the optimization of schedule are presented in [EPR95] and [Modi03].

A very interesting example on how team scheduling approach is realized in practice may be found in [EPR95], where a tractable heuristic algorithm for maximizing global utility through local plan combination is presented.

5.3.2 Advantages

Actually, the result of the team scheduling approaches is quite the same as a result of hierarchical approach – it is a common for all the agents schedule. But under team scheduling approaches the schedule is obtained in a totally distributed manner, which leads more reliability, speed, minimal bottlenecks. And it is still optimal!

Besides long-term scheduling allows doing precise estimate when each task will be executed, and when exactly the global goal will be achieved. So, within multi-agent systems with predictable task execution time, team scheduling approach makes the work even more predictable.

5.3.3 Disadvantages

The first and the most important disadvantage is the fact that long-term scheduling demands on the ability of the agents to predict the time of their tasks' execution. Obviously otherwise the scheduling has no sense, because the whole schedule will break down after the first unexpected delay. However far from all the production environments have a property of precise predictability. So, in hardly predictable systems the team scheduling approach is inefficient.

The second key point is a usually huge communication load while adding new tasks into the plan. That means that it is better not to use this approach in systems, where task arrive dynamically.

The third and the last disadvantage comes from comparing team scheduling and hierarchical approaches. Indeed, the common schedule is a result in a both cases, but under first approach it is obtained in a distributed manner. That is basically an advantage. But, on the other hand, it also leads to a increasing of the communication load, which may be considered as an disadvantage.

5.4 Stochastic Distributed Scheduling Approaches

5.4.1 Description

The term *stochastic distributed scheduling* comes from [HoRa97]. It means that, in contrary to the team scheduling approaches, the decision as to which workpiece transportation task is to be selected is done at the moment or arrival of this task. Stochastic distributed scheduling strategies have common with multi-planning coordination approach, but more likely to be classified as negotiation.

Stochastic distributed scheduling works as follows:

1. After the arrival of a new task, the robot with the highest utility of doing this task is selected for execution of it.
2. If for some own reasons elected robot prefers to refuse to do the task, than the first paragraph is repeated.

The difference between the team scheduling and stochastic distributed scheduling approaches is the time, for which the activity is scheduled. Under team scheduling the schedule is done for a defined, relatively long period of time, whereas under stochastic distributed scheduling the activity is scheduled only for nearest, relatively short time.

Graphically stochastic distributed approaches can be described as follows (Figure 5.6):

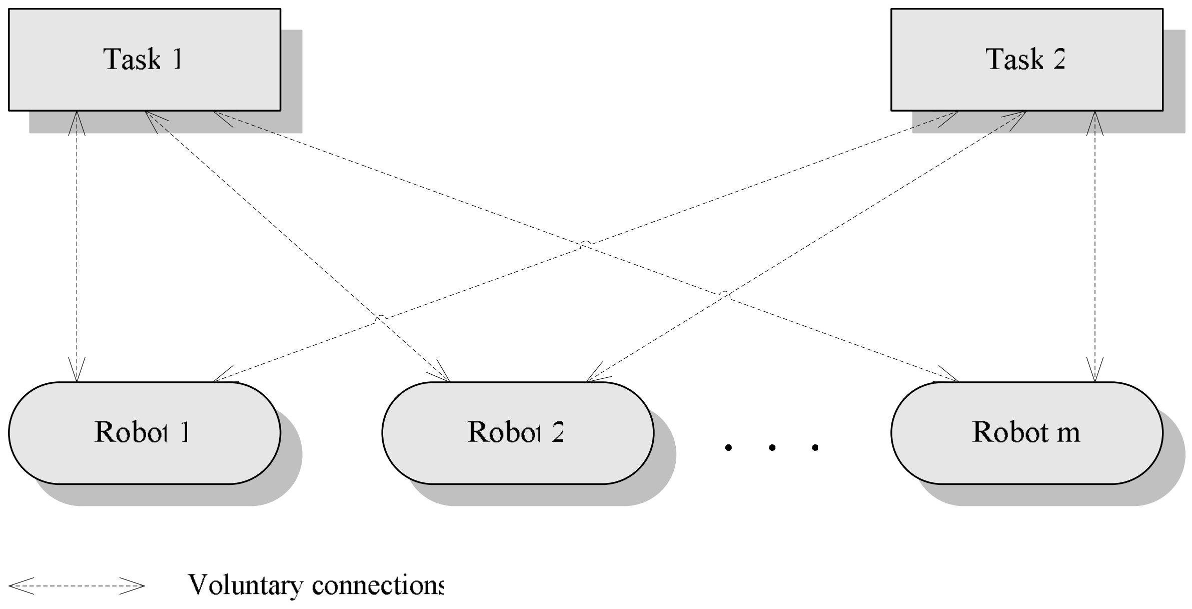

Figure 5.6: Stochastic distributed scheduling approach

A description of an approach for coordination of multi-agent systems using stochastic dynamical scheduling may be found in [LRP02]

5.4.2 Advantages

The stochastic distributed scheduling can be very useful in the systems where tasks arrive dynamically and should be executed also dynamically.

The absence of the long-term foregoing schedule protect system from the unexpected delays, such delays make no affect on the entire system at all.

5.4.3 Disadvantages

First of all, stochastic scheduling does not take into account the activity of agents that will be carried out even in the nearest future. So, it is worthless to suspect that obtained schedule will be optimal. However, in many real-word problems there is just no other way.

Secondly, under stochastic distributed problem solving process it becomes a real problem to get predictions how much time is needed to carry given task out. This issue can be critical for real-time systems that need to monitor the progress of the activity and predict the competitions time

6 Agent Oriented Analysis of the System

The central point of this project is the coordination and collaboration in multi-agents systems. So, the agent-oriented analysis and design of the system seem to be unreasonable. However obviously the development of coordination mechanisms is impossible without prior definition of units that behave actively and how they interact with each other. In other words, the roles in the system and the interactions between these roles should be defined before development

The agent oriented analysis and design for "Lego-robots" environment were already carried out in [Resc04]. According to this work, the following roles were defined:

- *Transportation* role (executed by robot agent)
- *Processing* role (executed by robot agent)
- *Placing and taking from storage* role (executed by special storage agent)
- *Time supervision* role (executed by special time supervision agent)

This approach is noteworthy but has a significant disadvantage: because of existence of two central roles (time supervision and work with storage) it is actually a centralized one. The centralized approach may be effective in small domains, however in bigger domains it has significant problem with speed and communication load to central agents. The second point is low reliability. Indeed, if one of the central agents fails then the whole system loses its ability to work.

That is why a new analysis of the system will be carried out with the aim of developing a completely decentralized model of the system. For this purpose the Gaia methodology will be used. It is one of the simplest methodologies for developing closed domain multi-agent systems. The analysis according to Gaia methodology consists of the definition of roles and interaction models. That is what is needed, so in this section of the work the Gaia agent-oriented analysis of the system will be carried out.

6.1 Elaboration of the Roles Model

The first and the main role in the flexible production is the transportation of workpieces. This role will control robots, receive the tasks and choose the most appropriate of them. Then the executor of this role should find the best way between two points and bring the workpiece from starting point to other. So, the first role *Transporter* is identified. Multiple agents will play this role.

But who gives tasks to Transporter? There are two principally different choices: either one central unit gives the tasks out or holding of the workpieces transportation tasks is distributed. As was said before, the centralized approach is inefficient from the point of view of scalability, reliability and speed. That is why distributed approach is selected and for each workpiece the separate role is defined. So, the second role *Workpiece* is identified, each workpiece will play this role.

To find a way between two points the transporter must use a free track. It means that the information about free and occupied tracks has to be kept. As it was said several lines before, there can be two approaches: centralized and distributed one. Under the centralized approach the information about tracks is kept in one central unit, under the distributed one – in many distributed units, each for one conventional track. On the one hand the second approach requires more communication and computation resources. However because of distribution and parallelism systems, based on the distributed approach are more reliable, quick and scaleable. There are important properties for flexible production environment automation system, so the distributed approach will be selected. Thus, the third role, carried by, multiple agents, is the *Tracks* role.

The aim of the production environment is the processing of workpieces on the stations. So, some unit should keep the information, how much time needs each station for processing, which stations are busy and which are free, and where the stations are situated. As with the previous roles the distributed approach will be selected. So, the fourth role *Station* is identified, for each station the separate station role will be defined.

Thus, the following roles were identified:
- *Transporter* Role (multiple)
- *Workpiece* Role (multiple)
- *Track* Role (multiple)
- *Station* Role (multiple)

6.2 Elaboration of the Interactions Model

The interactions between the four proposed roles are presented below (Figure 6.1):

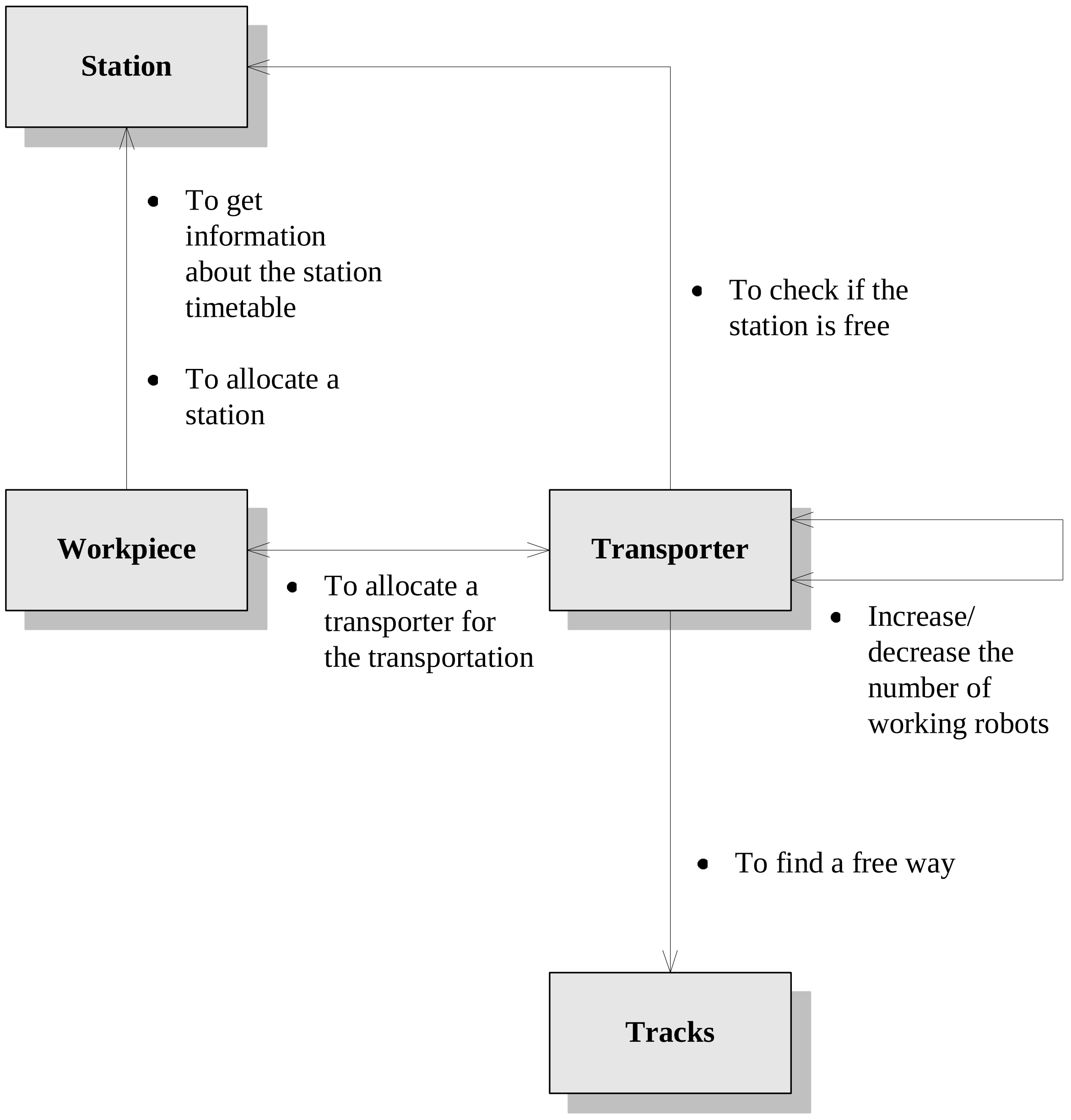
Station
To get information about the station timetable
To allocate a station
To check if the station is free
Workpiece
Transporter
To allocate a transporter for the transportation
Increase/ decrease the number of working robots
To find a free way
Tracks

Figure 6.1: Interactions between roles

Transporters, workpieces, stations and tracks interact with each other in the following way:

- *Workpieces with Stations* interaction. Before making a request to a transporter, a workpiece should first check the timetable of the station, and find the durations, when this station is ready for processing the workpiece. The second kind of interaction between station and workpiece initiates after finding a transporter that will execute the transportation task. In this case the workpiece allocates the station for specified time, needed for processing.
- *Workpieces with Transporters* interaction. After receiving the timetable from a station a workpiece makes a query to all transporters to carry out this task. In this query the workpiece also includes the stations timetable, so that the transporters could know when the station is free. On the answer transporters send the proposals – the utilities that would be achieved by each of them in case of execution of this task. The term “utility” defined the measure of success of performing a task. It will be explained in details later. Among these proposals the workpiece selects one with the highest utility, and send the request to execute this task to transporter that has proposed it. If request is successful, then the workpiece allocates the

station. From it’s part a transporter may reject or cancel the request, if at the time when request come it have another request with lower utility.
- *Transporters with Stations* interaction. Before coming to a station a transporter should check one more time if this station is free. The station has to be free because it was allocated by the workpiece, but it may happen, that because of some other transporter’s delay the schedule broke and the station is still occupied. In this case the Transporter should update it’s schedule and wait till the station became free.
- *Transporters with Track* interaction. Normally more than one Transporter is moving in the working field. That is why Transporter should coordinate the way they are moving on to avoid collision. To do that the entire working field is divided into defined parts, so-called tracks. Transporters should contact tracks before they move to them. If a needed track is occupied, then a Transporter should wait till the track becomes free, or find another track. If a needed track is free, then a Transporter should send a request to this track and occupy it. After leaving the track on which the Transporter was before this track should be released. This is also part of interaction between Transporters and tracks.
- *Transporters with transporters*. Usually there is no need for all the available transporters to be occupied. Some of the transporters may be free and join the work when it is needed. For that transporters constantly interact with each other with the aim of dynamical analysis of the total load, i.e. the load of all the system. If global load is too high, then the amount of robots should be increased. For that robots make a decision, which of the free transporters will join. In the same manner in case of too low load transporters s decide to decrease the amount of working transporters and make the decision which or the transporters will became free.

Thus, the roles and interactions model for flexible production environment were presented. These models are rather simplified and abstract, and before design of the agent-oriented system the more precise analysis should be carried of. However the level of complexity of proposed models is just enough to develop the coordination mechanism.

7 Development of Activity Selection Mechanism

To select the activity, agents need first to value the benefit of the each kind of activity. Which activity selection mechanism should be implemented in multi-agent automation systems?

There are two most appropriate types of agent, depending on the way how agents make a decision [RuNo03]. They are:
- Goal-based agents
- Utility-based agents

Goal-based agents have a description of desirable situation and try to reach this situation. Utility-based agents have a utility function, which describes each possible situation and represent the degree of success. On each step agents try to maximize this degree of success, by this maximizing the utility.

The goal-based approach is better to use in rather simple systems, where agents have only one goal, and try to reach it. In automation systems for flexible production environment this approach cannot work in an adequate way, because agents in this system (either workpiece or robot agents) have several choices what to do, and in most cases each of this choices is not ideal one. So agent should select not the desired one among remaining irrelevant actions, but the best one among other relevant actions. In other words, several actions are worthwhile to execute, but some of them bring a higher degree of success, and others a lower one. Obviously, in such conditions the utility based approach is the most appropriate one.

Thus, the utility-based approach will be used in the developed multi-agent automation system.

7.1 Execution of concurrent goals

In many real-world systems agents do not have only one parameter, that describes the state and according to which the utility is calculated. On the contrary, it can be several parameters, like, for example, optimization of transport costs, satisfaction of real-time requirements, etc.

In general, the execution of activity selection process may be divided into several steps:

- First, the *global task* is subdivided to several *subtasks*. In our production systems it may be the transportation of workpieces between stations.
- Then, each task is described according to several different parameters. Each parameter forms the corresponding *partial utility*. So, as many parameters, so many partial utilities.
- After that the partial utilities are being combined, and form the utility function for this subtasks. This utility describes the degree of success of the subtask.
- The final step (if it is needed) is to combine them to the global utility.

These transformations may be graphically represented like shown below (Figure 7.1).

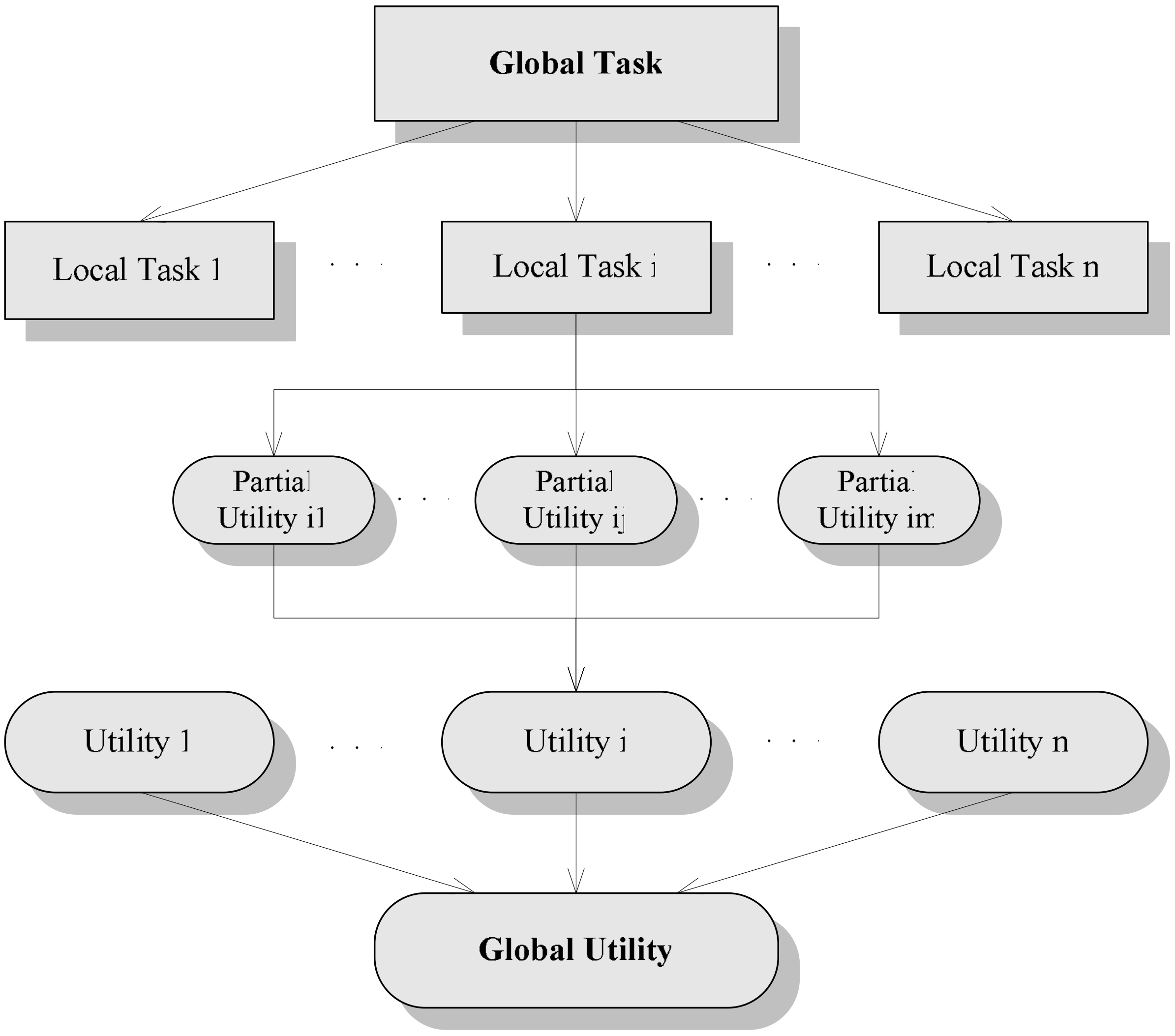

Figure 7.1: Transformations from global task to global utility

Partial utilities are real numbers, which characterize the degree of success of each task from the point of view of one of the definite parameters. In the “Lego-robots environment” it may be following partial utilities:

- *Transport optimization* utility
- *Soft Real-time* utility
- *Time Optimization* utility
- *Hard Real-time* utility

In the following subsections detailed information about how to represent each type of the partial utilities and how to combine them to the utility of the task.

In particular, two approaches are proposed:

- *Utility Based Approach*
- *Utility/Constraints Based Approach*

7.2 Utility Based Approach

Under the utility based approaches each partial utility is represented as a real number. The highest number means the highest utility. Then, the partial utilities are combined with the help of weighted sum, according to the formula:

$U = k_1 \cdot u_{p1} + k_2 \cdot u_{p2} + ... + k_n \cdot u_{pn}$,

where U is an utility, u_{pi} is a partial utility of type i and k_i is a coefficient for i-th type of partial utility (Figure 7.2).

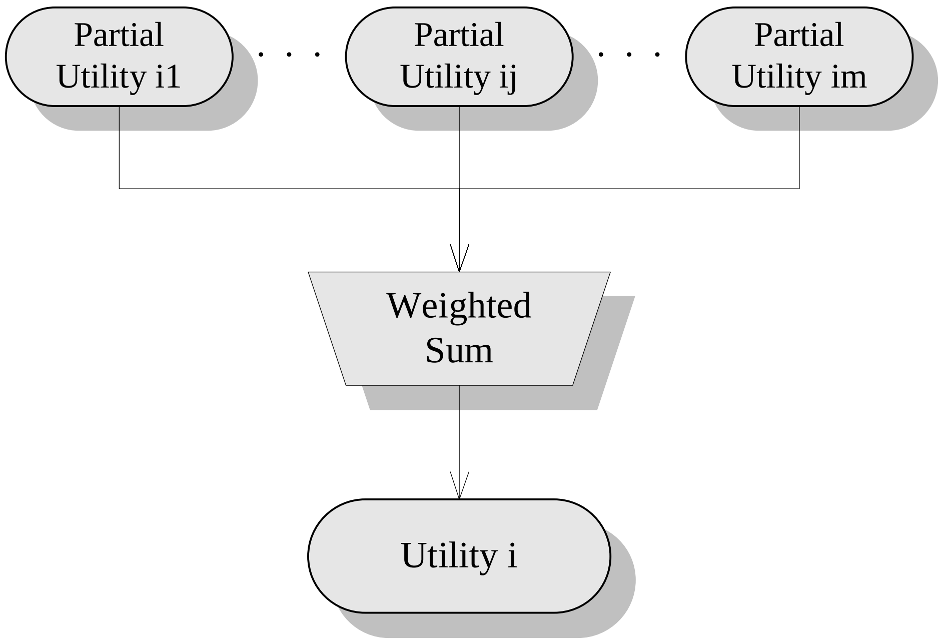

Figure 7.2: Combination of partial utilities to the utility

This approach gives the possibility to assign desired priorities to each type of the utility. For example, in real-time systems the real-time requirements utilities may have a higher coefficient than transport optimization utilities. On the contrary, in systems where time requirement are not very important, the coefficient for the transport optimization utility may be much higher, then the ones for real-time or time-optimization utilities.

Besides different partial utilities may have different measurement units. So, another advantage of combining partial utilities with the help of weighted sum is the possibility to normalize the partial utilities with different measurement units.

But how may different types of partial utilities may be represented with the help of a real number? The following subsections answer this question.

7.2.1 Transport Optimization Utility

Transport operations require resources, like energy, time, etc. Thus, it costs. The transport optimization utility must estimate these costs, and give the higher utility to the tasks that have lower cost and lower utility to those tasks that cost more.

For the “Lego-robots” system the formulation of transport optimization utility is rather simple, because in general we should not consider about energy because all robots are equal, thus they consume equal energy for equal terms. On the same reason the amortization parameter also should not be considered.

The only parameter that describes the transport costs is time, needed for transportation. In general the whole *task time* may be divided to the following parts:

- *Waiting time* is the time, that passes before beginning of the task execution.
- *Free-ride time* is the time, that robot needs to come to the station where the workpiece is situated.
- *Benefit time* is the time, when workpiece is being actually transported.
- *Execution time* is the time that robot moves, so it the sum of the free-ride time and the benefit time.

These times follow each other as it is shown below (Figure 7.3):

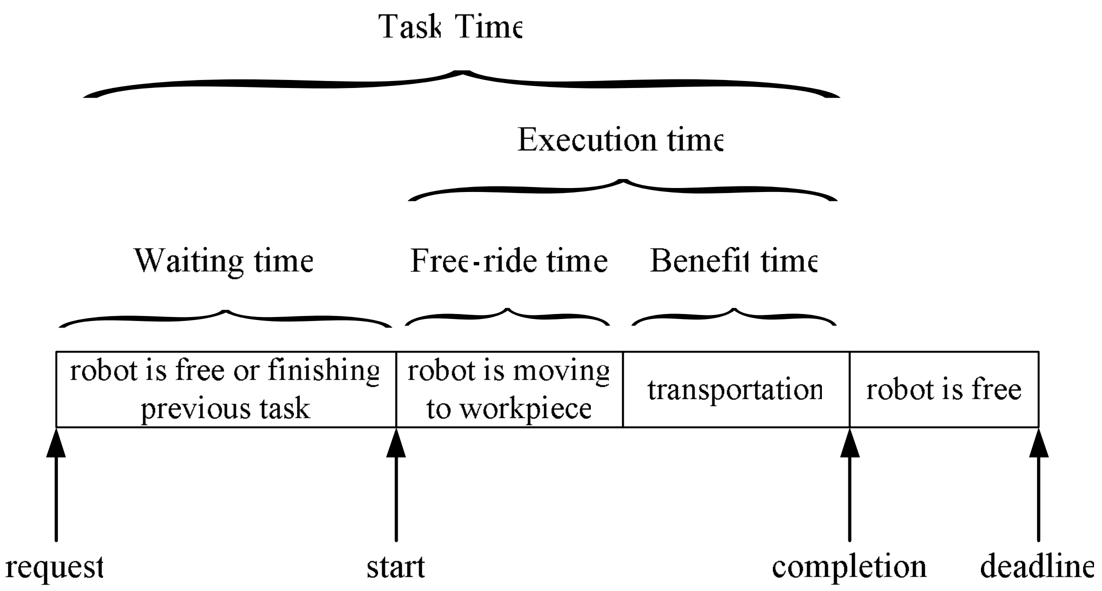

Figure 7.3: Transportation times

The aim of each agent is to spend as less time as possible on the task. So, the partial utility function may be represented in the way like on the figure below (Figure 7.4).

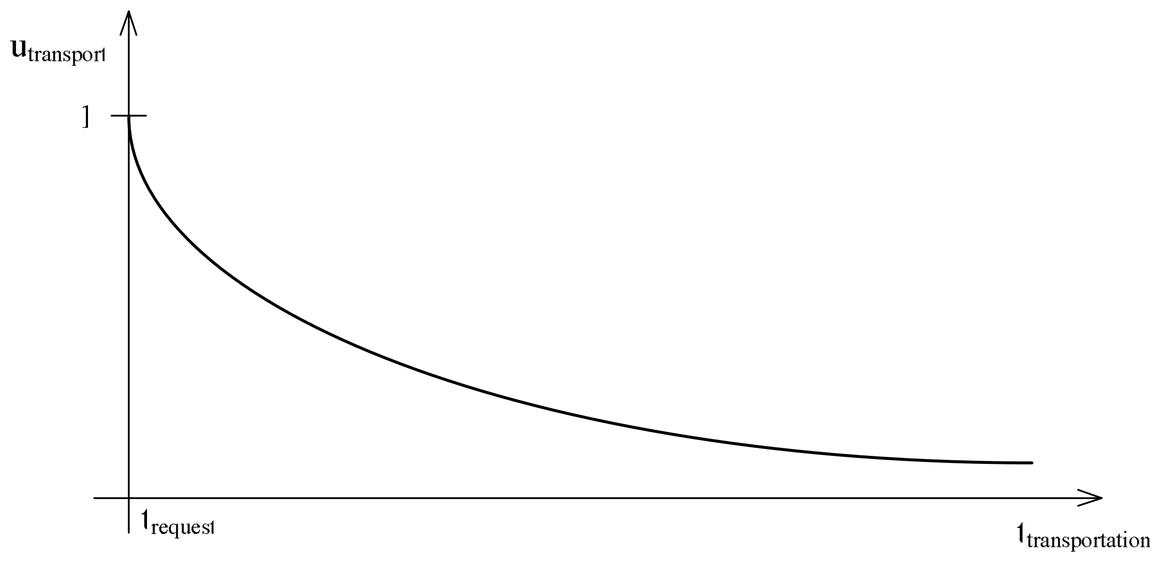

Figure 7.4: Transport optimization partial utility

The proposed graph may be represented with following function:

$$u_{transport} = e^{-t_{trasportation}}$$

7.2.2 Soft Real-Time Requirements Utility

In real-time systems tasks should not be only executed, they also should be completed till the defined times, so-called *deadlines.*

There can be two kinds of real-time systems:
- *Soft real-time* systems
- *Hard real-time* systems

In hard real-time systems strict deadlines must not be missed in any case [Göhn04], otherwise the task brings no utility. In soft real-time a violation of deadlines can be tolerated, in utility notion it means that after the deadline the utility decreases.

Thus, the partial utility function for soft-real systems has the following view [Göhn04] (Figure 7.5):

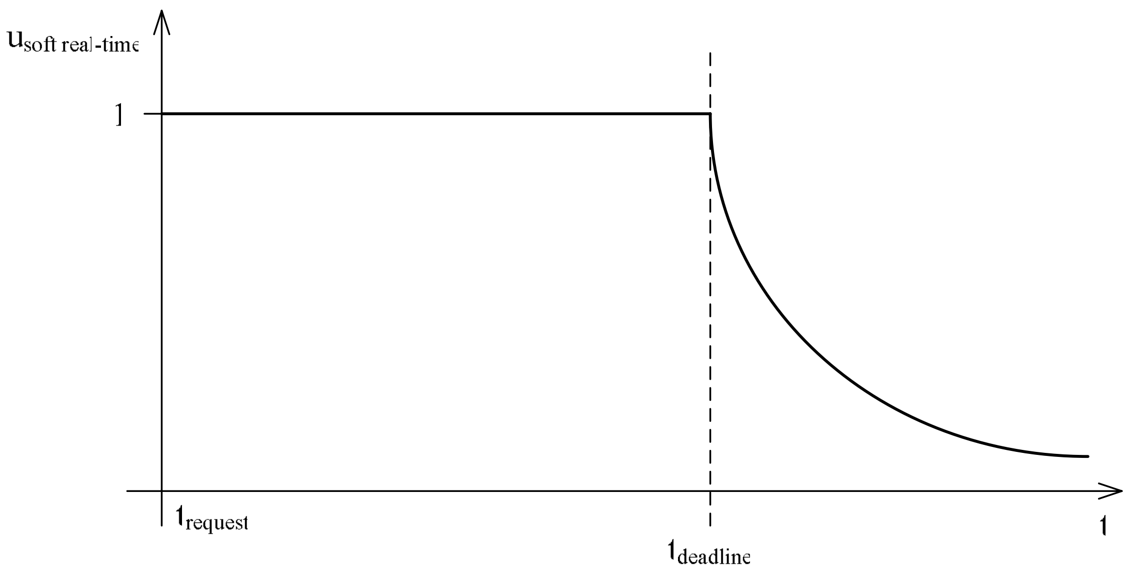

Figure 7.5: Soft real-time requirements partial utility

It may be described by function:

$$u_{soft\ real-time} = \begin{cases} 1, & t_{completition} \le t_{deadline} \\ e^{-(t_{completition} - t_{deadline})}, & t_{completition} > t_{deadline} \end{cases}$$

7.2.3 Time Optimization Utility

The soft real-time utility, proposed in the previous subsection, brings two problems:

- The predicted execution time is not always correct because the robots may execute tasks with delay. So, the laxity time should be maximized, or in other words the tasks with small laxity time should be preferred to those with big laxity time.
- According to the proposed utility function the tasks with passed deadline will benefit a low utility. That means, that no robot will prefer to take such a task. Obviously that is wrong, quite the contrary the tasks with passed deadline should have the highest preference.

The graph of the partial utility function that correspond to the above described two problems, namely the maximization of laxity time and high preference to task with passed deadline, is presented on the figure below (Figure 7.6):

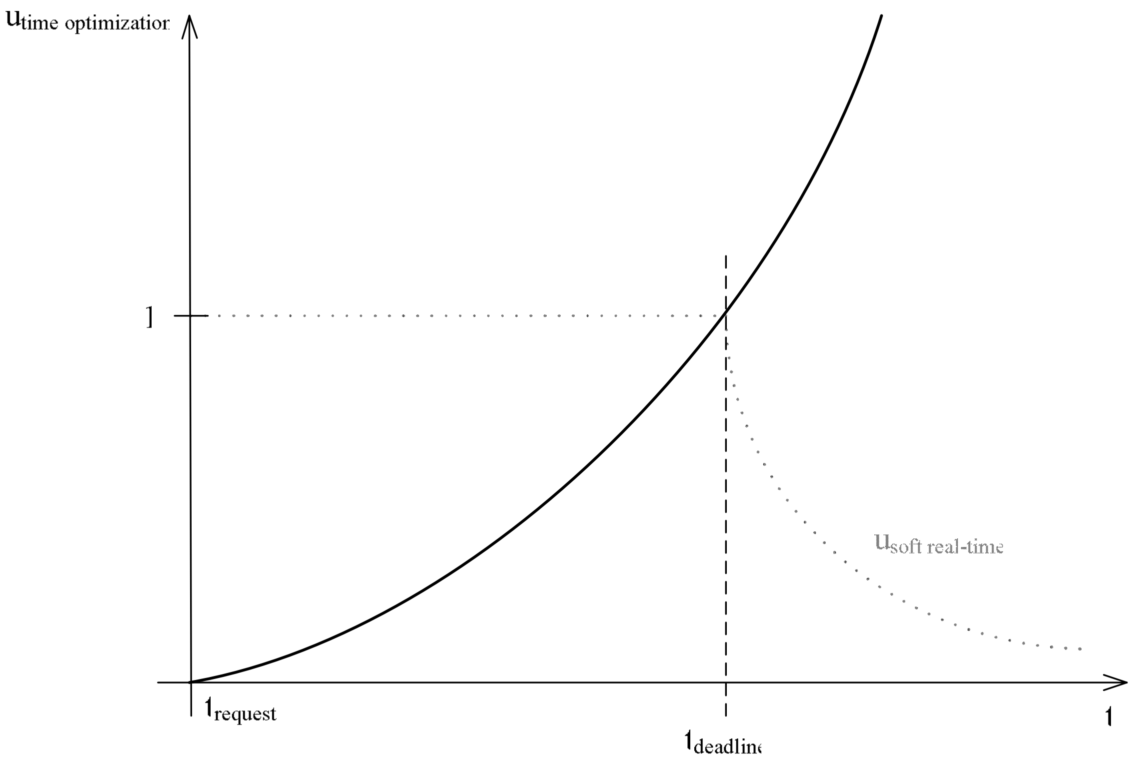

Figure 7.6: Time optimization partial utility

This graph may be described with the following function:

$$u_{time\ optimization} = \left(\frac{t - t_{request}}{t_{deadline} - t_{request}} \right)^p ,$$

where *p* is predefined coefficient. The higher this coefficient is, the more preference tasks with small laxity time and tasks with passed deadline have.

7.2.4 Hard Real-Time Requirements Utility

The explanation of hard-real time requirements partial utility is analogous to the explanation of soft real-time requirements partial utility. The only difference is that in hard real-time systems the utility of the tasks after the deadline equals zero. So, the graph of the partial utility function will have the following appearance (Figure 7.7):

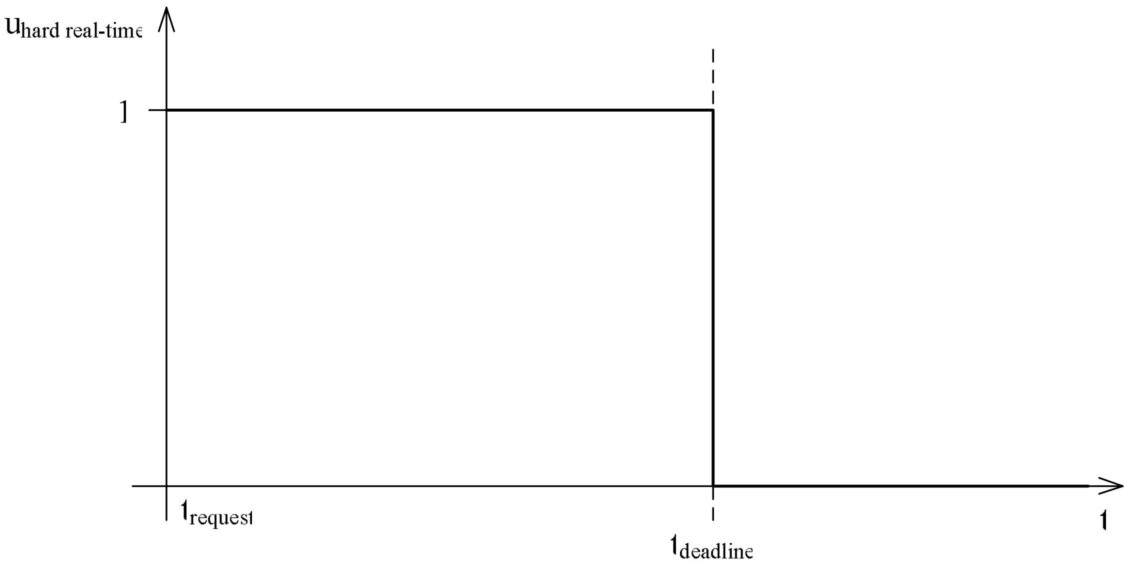

Figure 7.7: Hard real-time requirements partial utility

This graph may be described with the function:

$$u_{hard\ real-time} = \begin{cases} 1, & t_{completition} \le t_{deadline} \\ 0, & t_{completition} > t_{deadline} \end{cases}$$

7.2.5 Summary

Thus, in the preceding subsections the partial utility function for 4 parameters that characterize tasks were proposed. Now, according to this, the utility function of the task for soft real-time systems looks like follows:

$$U = k_{transport} \cdot u_{transport} + k_{soft\ real-time} \cdot u_{soft\ real-time} + k_{time\ optimiyation} \cdot u_{time\ optimiyation}$$

And of hard real-time as follows:

$$U = k_{transport} \cdot u_{transport} + k_{hard\ real-time} \cdot u_{hard\ real-time} + k_{time\ optimiyation} \cdot u_{time\ optimiyation}$$

It is worthwhile to mention, that the proposed partial utility functions were defined completely empirically, according to properties of the “Lego-robots” system. Coefficients of weighted sum should be defined according to the requirements of the developed system.

In general, the definition of utility functions is an engineering task that should be carried out for every individual system, according to the requirements of the system to be developed. The proposed 4 partial utilities are only an example, how different parameters may be combined into one utility. In the concrete developed system this representations may differ and other partial utilities may be added.

7.3 Utility/Constraints Based Approach

Let’s look what happens in the hard real-time system after the deadline was passed. Obviously there is no sense to execute this task any more, because in hard real-time systems the utility of the task after deadline equals zero. But how copes with this utility based approach? After the dead-line the hard real-time requirements partial utility equals zero, but other term of the weighted sum are not! They are bigger than zero, so the utility of the task will not be equal zero.

Thus, the utility approach acts inadequate with the hard real-time requirements partial utility. In general, it acts inadequate with all partial utilities that coming to zero should make the utility of the whole task also zero.

There may be several possibilities how to resolve this problem. Two of them are:

- To use multiplication instead of weighted sum. In case of multiplication when at least one partial utility equals zero, the utility of the task also equals zero. However, this approach has an important disadvantage: not all of the partial utilities must turn the utility of the task to zero after turning to zero themselves. With multiplication they do.
- To use the utility/constraints approach. Under utility/constraints approach all of the parameters may be described like partial utility *or* constraints (Figure 7.8).

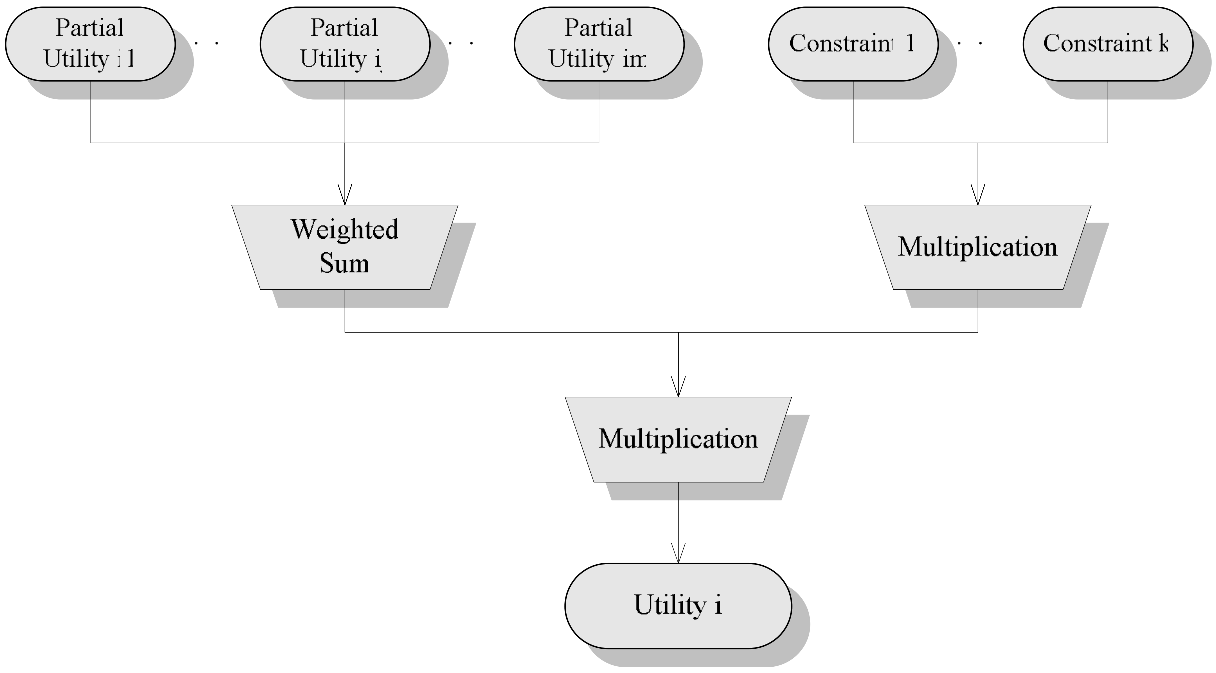

Figure 7.8: Utility/constraints based approach

Partial utilities are, as earlier, a real number that describes the degree of success from the point of view of a defined parameter. Constraints are binary numbers, that also characterize the task, but may have only two values: 1 or 0. The partial utilities and constraint are combined into utility according to the following formula:

$$U = (k_1 \cdot u_{p1} + k_2 \cdot u_{p2} + ... + k_n \cdot u_{pn}) \cdot (c_1 \cdot c_2 \cdot ... \cdot c_n)$$

So if one of the constraints equal zero, then the whole utility of the tasks is zero. Otherwise the utility is defined as under the utility approach using the partial utilities.

So, the hard-real time requirements may be represented as a constraint:

$$c_{hard\ real-time} = \begin{cases} 1, & t_{completition} \leq t_{deadline} \\ 0, & t_{completition} > t_{deadline} \end{cases}$$

or, in graphical form (…):

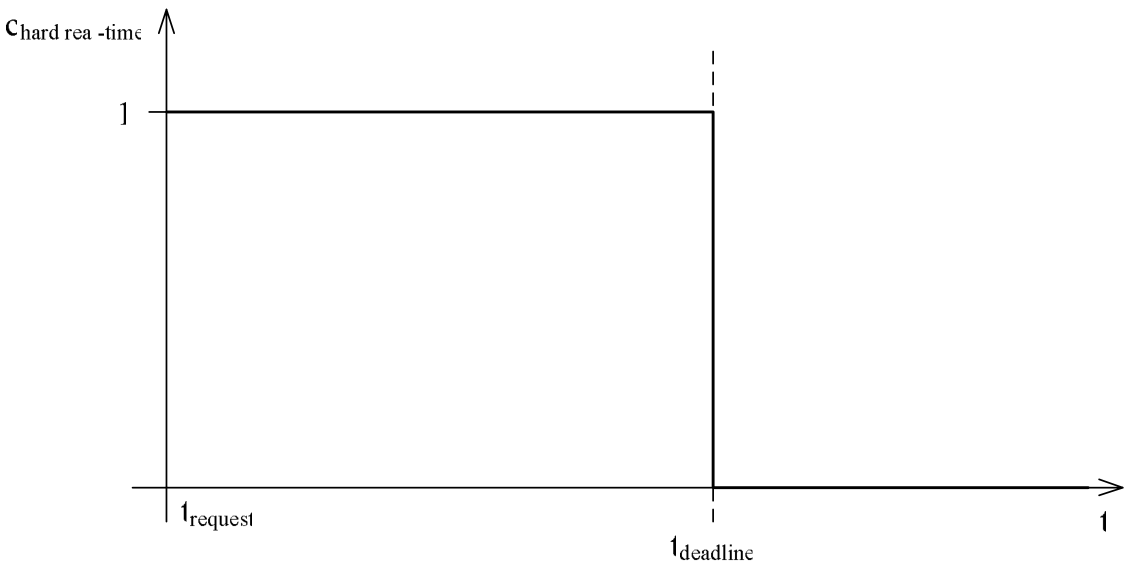

Figure 7.9: Hard real-time requirements constraint

The formula for calculating the utility under the utility/constraints based approach is:

$$U = \left(k_{transport} \cdot u_{transport} + k_{time\ optimiyation} \cdot u_{time\ optimiyation}\right) \cdot c_{hard\ real-time}$$

Thus, the utility/constraints based approach copes with constraint-like parameters that are binary inherently and mean, if the task will have success at all, or not. This approach interprets the constraint as a factor to weighed sum. In all other aspects utility/constraint based approach is analogous to the utility based approach

8 Development of a Coordination Mechanism

So, up to this moment roles and interaction models are already developed. The roles model gives information which roles are present in the system and interactions model defines how this roles interact with each other.

In this chapter the development of the coordination technique for the flexible production environment will be carried out. First the advantages and disadvantages of selection of different approaches will be analyzed. Then a coordination technique for the “Lego-robots” flexible production environment will be elaborated.

8.1 Analysis of Possible Approaches

8.1.1 Approaches Overview

As it was identified before, there may be 4 approaches to coordination in MAS for flexible production environment:

- Hierarchical (centralized);
- Contract net protocol based;
- Team scheduling;
- Stochastic distributed scheduling.

Despite its simplicity and ability to generate efficient schedules with low communication load, *the hierarchical (centralized) approach* has all the disadvantaged typical to all centralized approaches. Namely it is low reliability (if the central agent fails, then the whole system fails), low scalability (with the increase of number of working agents the requirements to central agent may become unrealizable) and low working speed (also because of the central agent, that must take part in all interactions). That is why this approach is unacceptable for the flexible production environments automation systems.

On the contrary *the contract net protocol based approach* is decentralized by its nature which increases the reliability and speed. However this approach requires too much activity from the robots, in fact under this approach both the roles transporter and workpiece are played by robot agent. It brings a high degree of unevenness to the load of the system - robots must do too many activities: to control moving, to receive tasks, to interchange tasks with each other. Moreover it affects reliability: if a robot agent with already assigned task fails, then at once the transportation tasks of several workpieces will be lost. The third disadvantage is the scalability of a system that implements the contract net protocol based approach. With growth or robots the communication load increases greatly, which leads to the inability of scaling. So, the contract protocol based approach to coordination is also unacceptable for flexible production environments like the model process "Lego robots".

The last two coordination approaches, *team scheduling* and *stochastic distributed scheduling*, are very similar. In both of these approaches the task is assigned to a transporter during the interaction between the workpiece and the transporter roles. Evidently it will be two different agents, for example workpiece and robot agents. So, the structure of multi-agent system is completely distributed, which affects positively to the reliability, flexibility, working speed and scalability. The main disadvantage is the relatively high communication load, needed for task allocation. However this load is distributed between all the agents, so it doesn't bring a significant problem even in the case of big domains.

The difference between these two approaches is the time interval, for which the activity is planned. Under team scheduling approach the activity for *all known* transportation tasks is planned, and under stochastic distributed scheduling approach the activity only for transportation tasks, *that should be executed in the immediate future*, is planned. In other words, under team scheduling the request comes in advance, earlier than coming of start time. Under stochastic distributed scheduling the request and start times contemporize (Figure 8.1)

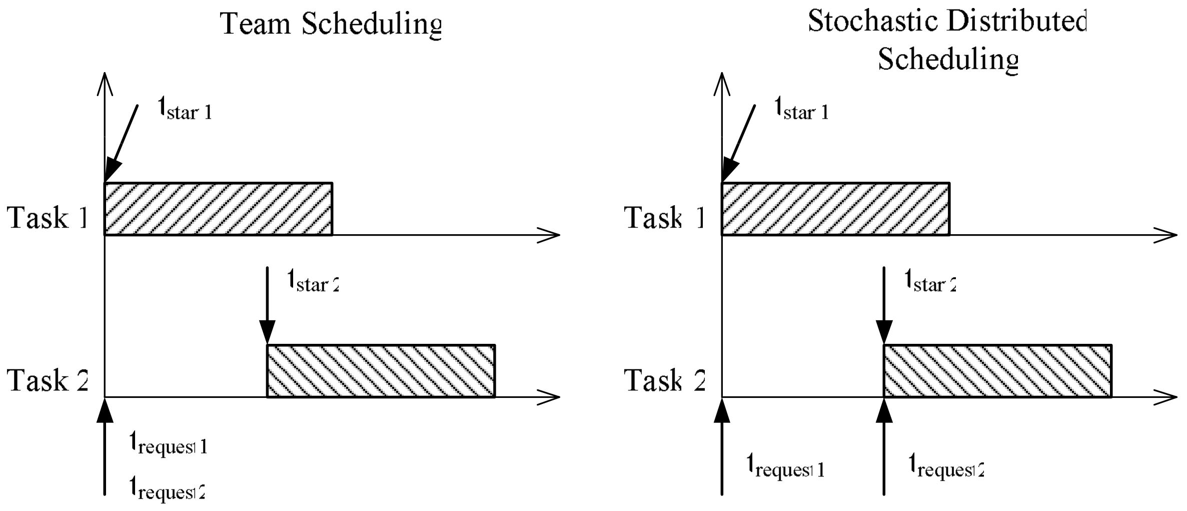

Figure 8.1: Team scheduling and stochastic distributed scheduling approaches

Under team scheduling approach all tasks are known in advance and scheduling is done for all the tasks. The schedule is elaborated only once. Rescheduling is possible operation; however it is undesirable because of a high communication and computation load, so it is used only in emergency cases. First phase is to do the initial mapping of tasks and transporters, which is done according to the utilities of the tasks. In the second phase the optimization follows, under this phase both the tasks and the transporters are trying to maximize the global utility. For that transporters may cancel the task, if another task with higher utility comes. In that case the cancelled task tries to find another transporter.

On the contrary under stochastic distributed scheduling no activity is planned in advance, there is no schedule for future activities at all. The requests are coming in the start time, and the decision is done immediately, which transport is to be selected. No transporter is allowed to cancel the task in case of coming a new task, because the task is being started right after the request.

8.1.2 Advantages of team scheduling and disadvantages of stochastic distributed scheduling

In general, the team scheduling approach gives more accurate and precise schedules. Below follow two simple examples, which show how systems work under stochastic distributed scheduling approach and how team scheduling approach can improve the work of the system:

- *Execution of territorially distributed tasks*

Let us assume following situation: the environment consists of 2 robots, 2 stations and storage. One of the robots, robot 1, is located on the line between station 1 and station 2, but close to station 1. Let us assume, that the distance between station 1 and station 2 is *4x*, station and robot 1 is *1x*, and between station 2 and robot 1 is *3x*. Robot 1 is located right down from station 1, and the distance between it and station 1 is *2x* (Figure 8.2).

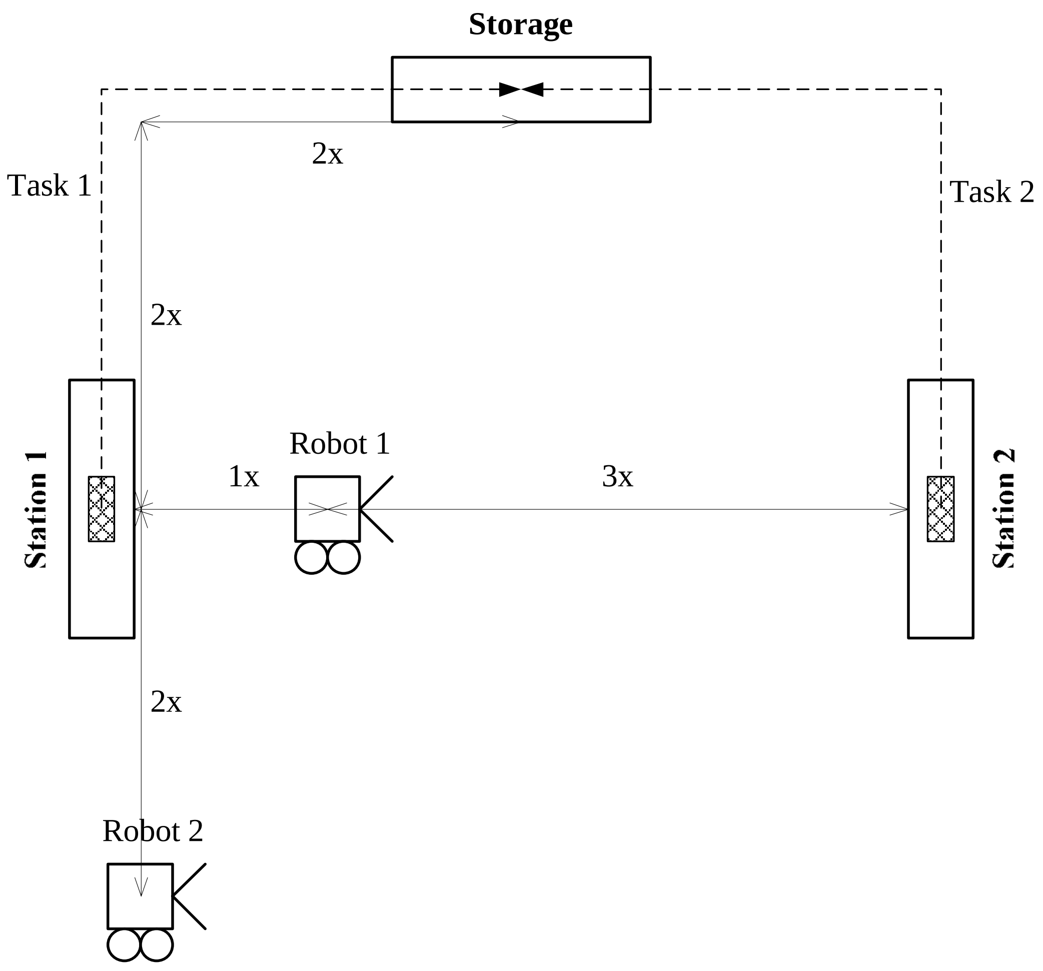

Figure 8.2: Environment of example 1

Two transportation tasks are given. Task 1 is to transport a workpiece from station 1 to the storage. Task 2 is to transport a workpiece from station 2 to the storage. Task 1 comes at time t_1, and task 2 comes at t_2, and $t_1 < t_2$ (Figure 8.3).

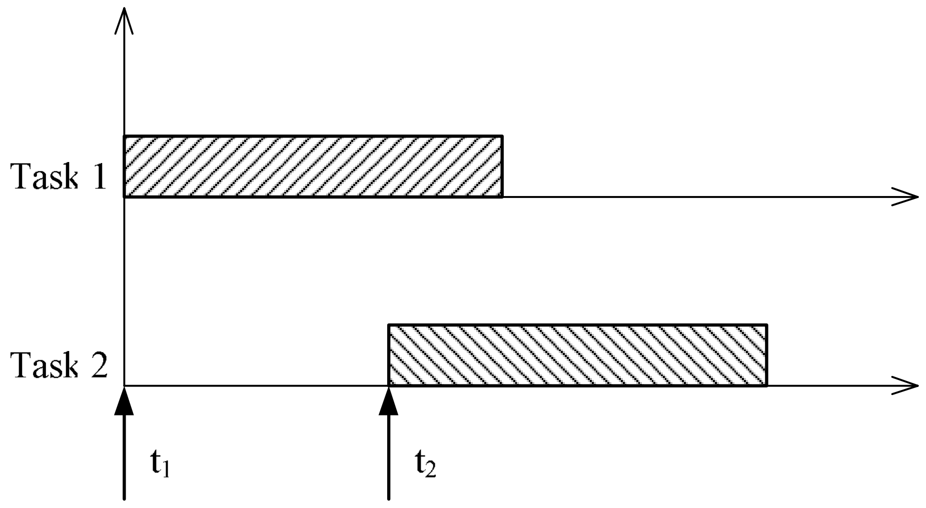

Figure 8.3: Time parameters of the tasks for example 1

Let us also assume that the only optimization criterion is the distance, passed by robots.

Now let's analyze how the system works under different approaches. Under stochastic distributed scheduling task 1 comes first. Obviously Robot 1 is closer to station 1, so it would be selected for executing this task. The total distance, passed by robot 1, in that case would be *1x + 4x = 5x*. Then the task 2 comes. At that moment robot 1 is already busy, so the task will be assigned to robot 2. For executing this task robot 2 should cover a distance *6x + 4x = 10x*. In sum, under a stochastic distributed scheduling the distance covered by both robots is *5x + 10x = 15x*.

Under team scheduling approach during the optimization all the possible variants are compared, and the best among them is selected. So, let's compute the second task allocation, namely when task 1 is assigned to robot 2, and task 2 is assigned to robot 1. In that case the costs for transportation of task 1 will be *2x + 4x = 6x*, and for transportation of task 2 – *3x + 4x = 7x. In sum - 6x + 7x = 13x.*

So, it is quite clear, that the second allocation is better, but under stochastic distributed scheduling it would not be recognized. It happens because under stochastic distributed scheduling no tasks are analyzed in advance.

- *Execution of tasks with near deadline.*

Another typical example is the following. Let us assume that the environment consists of robots and two tasks. That means that the tasks should be executed consecutively, the only question is which the first is.

Task 1 comes at time t_1, and task 2 comes at t_2, and $t_1 < t_2$. The execution time of task 1 is much longer, then of task 2, but it also has a more distant deadline (Figure 8.4).

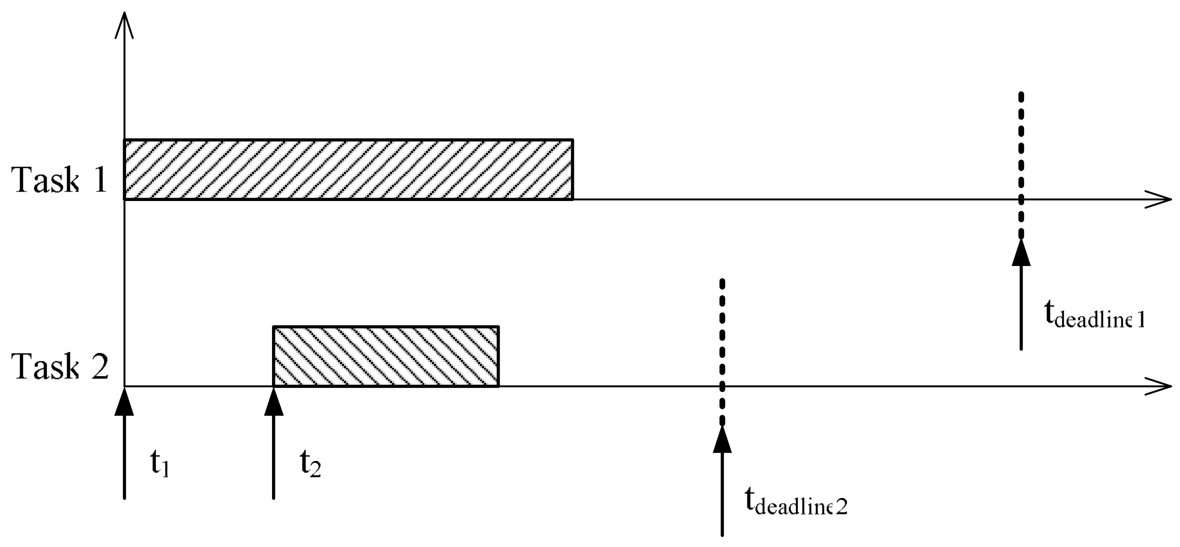

Figure 8.4: Time parameters of the tasks for example 2

It is quite clear that robot should wait for first task 2 coming, then execute it, and after that execute task 1. Otherwise, if task 1 will be executed first, the task 2 will be completed after its deadline. That will be detected under team scheduling approach and task 2 will be executed first. But under stochastic distributed scheduling at the request time of task 1 nothing is known about task 2, so the robot starts executing task 1. Then comes task 2, but it is too late to change anything, so robot will first complete with task 1, and only then execute task 2. Obviously the task 2 will be then completed after deadline.

Thus, as in the previous example team scheduling approach shows its ability to make better schedules, because tasks are analyzed in advance.

8.1.3 Disadvantages of team scheduling and advantages of stochastic distributed scheduling

So, as it was shown above, the team scheduling approach gives better results thanks to prior analysis of time parameters of the tasks and optimizing the schedule according to these parameters.

But what happens if the prior precise information about time parameters of tasks is not available? Let us examine following situation like in the second example in the previous chapter, but let us assume that we don’t know the precise time, when task 2 will start and when both task will be completed (Figure 8.5).

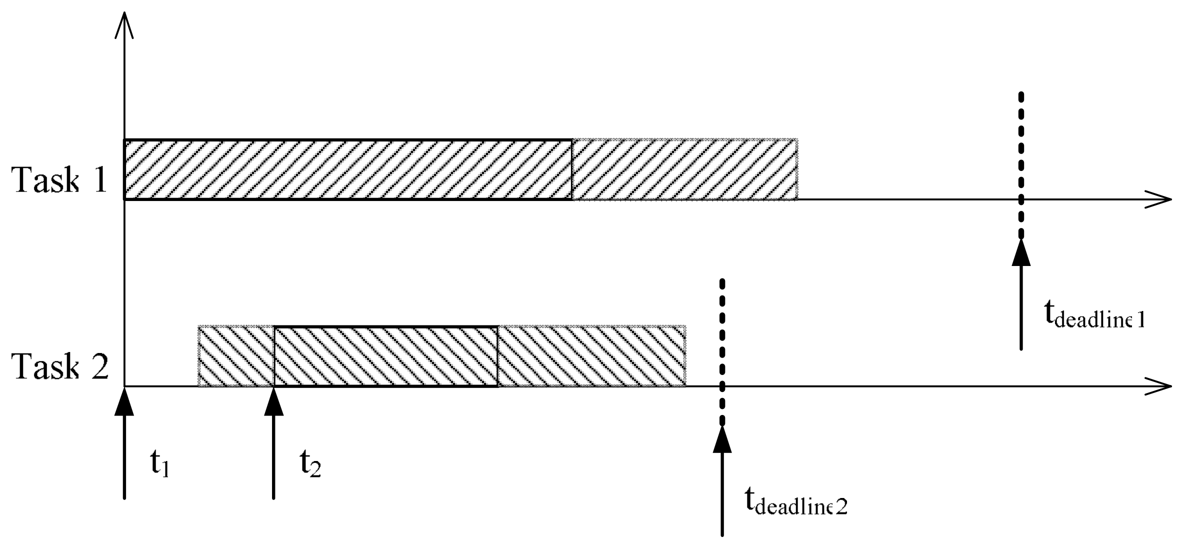

Figure 8.5: Time parameters without precise request and completion time

One possible decision is to make approximate estimation of start and completion time, define allowable difference between estimated and real values of these parameters, and make aschedule based on these parameters. Then in every case when the real differences are not in the allowable limits the rescheduling should be done. But under team scheduling the scheduling is done for long terms, which means that rescheduling is very communication- and computation-expensive operation. Obviously that is an absolutely ineffective approach.

That is why the team scheduling does its best only in systems with the possibility of precise estimation of time parameters. In other systems it is much better to use stochastic distributed scheduling approach, because anyway in such systems team scheduling doesn't bring much profit, comparing with stochastic distributed scheduling, but stochastic distributed scheduling need much lower computation and communication resources.

So, team scheduling is an efficient approach for systems with possibility of prior precise estimation of the time parameters, so-called predictable systems, because it allows creating schedules that take into account all the tasks. However In other systems it is much better to use stochastic distributed scheduling, because this approach is completely dynamic and doesn't foresee a long-term scheduling, so it doesn't need to do a rescheduling because of the delays.

8.2 Elaboration of Coordination Technique

In this subsection the coordination technique for flexible production environment on the example of process model "Lego-robots" is elaborated.

In the previous subchapter it was stated that the most effective coordination approaches for distributed flexible production environment automation system is team scheduling and distributed stochastic scheduling. The team scheduling approach is better for systems with predictability property, and stochastic distributed approach is better for more stochastic systems. What kind of system is flexible production environment?

Typically work under workpieces consists of cycles that consist of two actions:

- Processing of the workpiece on the station
- Transportation

First the workpiece is transported from the storage to a station. Then the workpiece is being processed on the station. Then the next transportation follows, and so on (Figure 8.6).

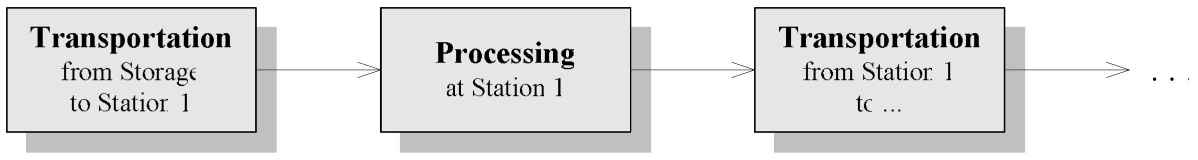

Figure 8.6: Workpiece transportations

From the point of view of time parameters processing of workpieces is deterministic. In other words, it is always known at the moment of processing start how long task will be processed, and hence when the processing will be completed and when the following transportation may be started. So, it gives possibility to use team scheduling approach that gives better schedules because of advance planning of the future activities beginning from the start time of processing one the station.

On the other hand the transportation is a completely indeterministic action, as the mechanical features of the robots make it impossible to precisely predict the execution time, because robots need a completion time of the previous task to plan next task.

Thus, none of the above described approaches may be directly implemented for the flexible production environment “Lego-robots”. Under team scheduling approach a lot of rescheduling will be needed because of the delays during transportations. On the other hand stochastic distributed scheduling will not use the predictability property of processing at the station, which decreases the efficiency.

Reasoning from above mentioned it is decided to develop a combination of the two approaches. This combination is *called stochastic team scheduling*. Under this approach the team scheduling is done, but only for those tasks, for which the precise time parameters are known. Typically that will be tasks that are being processed at the station or are waiting for robot to transport them. All other tasks should wait till their time parameters will be defined, only after that the planning for them begins.

So, the main condition for starting the scheduling process for the task is *the precise knowledge of it’s start time of the transportation.* When this information is known, then the request comes without waiting, before start time coming. After that, if there are free robots available, team scheduling process begins, otherwise the task waits for one of the robots to become free.

In that way, this new approach has following features from both approaches:

- From the team scheduling approach: the task execution for tasks with precise information about start time is planned in advance, which increases the efficiency of the schedule;
- From the stochastic distributed scheduling approach: the tasks is completing stochastically, hence the requests and rescheduling also occur stochastically.

The process under stochastic team scheduling is the following: first, like under team scheduling approach, the initial planning of the several tasks, for which time parameters is known, is done. Then, with the execution of these tasks, the time parameters of other tasks are stochastically ascertained, and for each of new tasks the team rescheduling process is also stochastically executed. To do the optimization under stochastic team scheduling approach agents are allowed to cancel the already assigned tasks. The important point is that because of a fewness of the tasks in the schedule the rescheduling is not a very expensive operation and it optimizes the schedule significantly.

In general, rescheduling under stochastic team scheduling is executed in two cases:

- when new task comes, i.e. information about time parameters of the task is ascertained and request to the transporter to execute this task is done;
- when transporter completes previous task and informs workpieces that the new task request may be accepted.

Thus, the stochastic team scheduling coordination approach inherits advantages from team scheduling and stochastic distributed scheduling approaches and uses coordination mechanisms depending on task parameters.

9 Elaboration of Evaluation Scenarios

Let us assume the environment that consists of three tasks and two robots and examine the coordination processes that occur in this environment under different coordination approaches, namely team scheduling, stochastic distributed scheduling and stochastic team scheduling.

Let task 1 and 2 have a small time interval between their start time and task 3 have to start some time later (Figure 9.1)

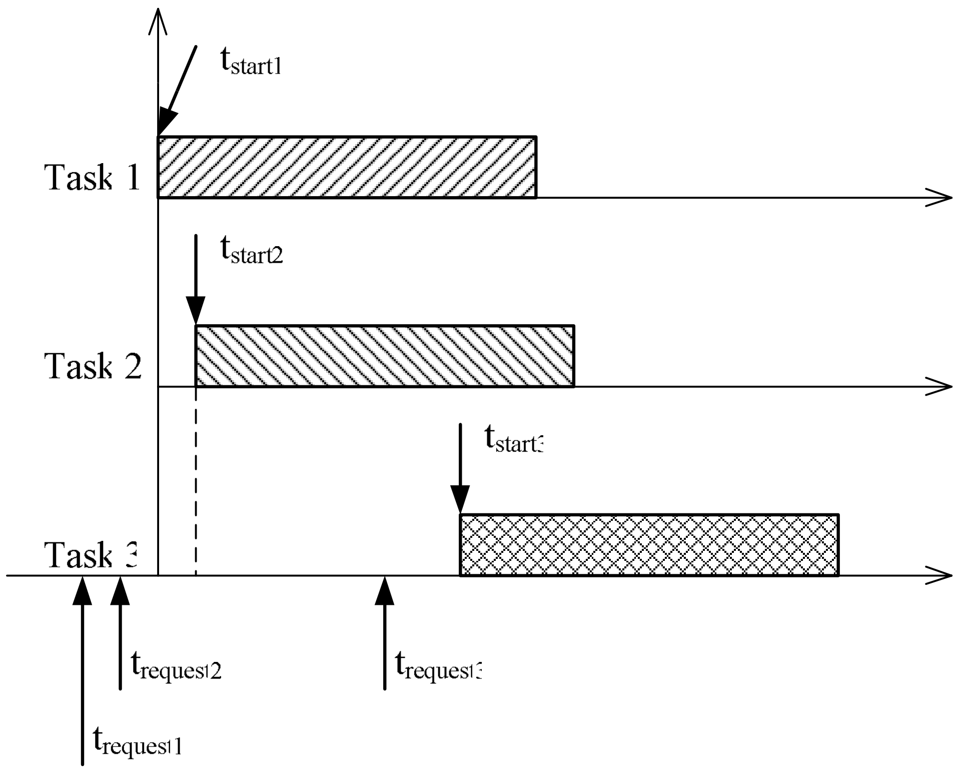

Figure 9.1: Time parameters of the example environment

Requests to the execution of the tasks, as it is typical for the “Lego-robots” flexible environment, come a little bit earlier than the start time, namely at the time when the workpiece has already been transported to a station and is being processed at the station at the moment. In such case the time of the processing completion is known and the request may be done.

The assumed utilities of the execution of each task on the different transporter are presented below (Table 9.1: The utility if the tasks

). For the simplification let the utilities not change with the time, i.e. the utility of the task does not change whenever it started.

Task Number	**Utility by execution on transporter 1**	**Utility by execution on transporter 2**
1	8	6
2	10	3
3	15	14

Table 9.1: The utility if the tasks

Task 1 has a higher utility if being executed on transporter 1, and lower if on transporter 2. On the contrary task 2 has a higher utility if being executed on transporter 2 and lower if on transporter 1. Task 3 has much higher utility on both transporters, comparing with previous tasks, but transporter 1 is a little bit more preferable for it.

9.1 Team Scheduling Approach

Under the team scheduling approach all the activities are planned in advance, before starting a global transportation process. No rescheduling in ordinary cases is foreseen. In other words, the precondition for application of team scheduling is the precise information about all the tasks, execution of which will be scheduled.

In the example presented above in advance it is known only about task 1 and task 2. These tasks will be planned. Obviously, as the utility of task 1 is higher if it is executed on transporter 2, and the utility of task 2, task 1 will be executed by transporter 2, and task 2 will be executed by transporter 1 (Figure 9.2)

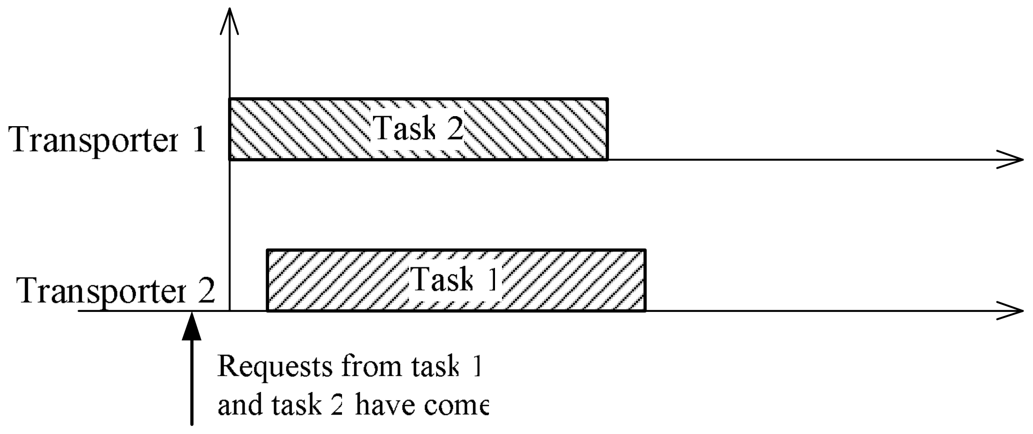

Figure 9.2: Scheduling under team scheduling approach

The request from the workpiece to execute task 3 comes later, after starting executing the tasks. As was said before, no rescheduling is foreseen, so, it won't be included in the schedule.

9.2 Stochastic Distributed Scheduling Approach

Under stochastic distributed scheduling the decision which task will be executed by which transporter is made at the time of request coming, and the transporters are not allowed to cancel the tasks.

Thus, the coordination between tasks will be carried out in the following stages:

1. *The request 1 from task 1 comes.* This request is received by all transporters and all transporters send the utility of this task in answer. The workpiece selects the transporter that brings the highest utility. Obviously it will be transporter 1 that brings utility 8. (Figure 9.3)

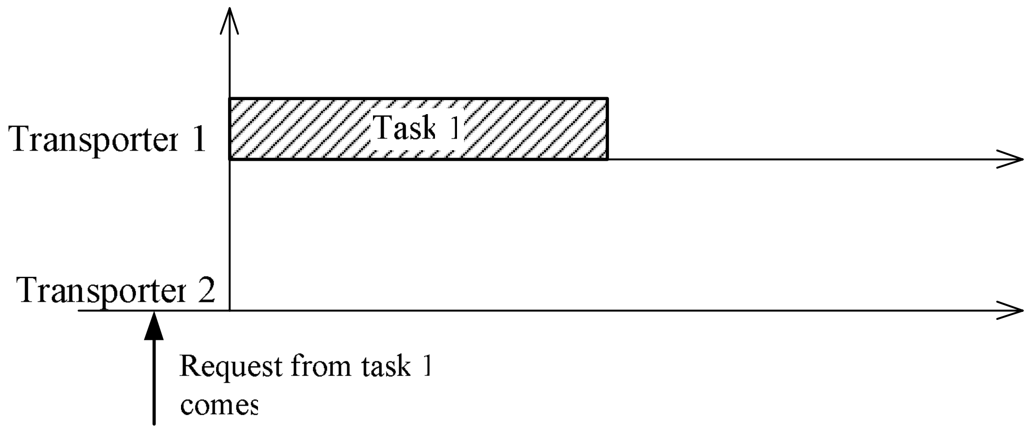

Figure 9.3: Stage 1 under stochastic distributed scheduling approach

2. *The request 2 from task 2 comes*. This request is received by all the transporters. Transporter 1 can give the highest utility, however it is already occupied, and it is unknown, when he will be free. So, task 2 will be assigned to transporter 2 that brings utility 3 (Figure 9.4).

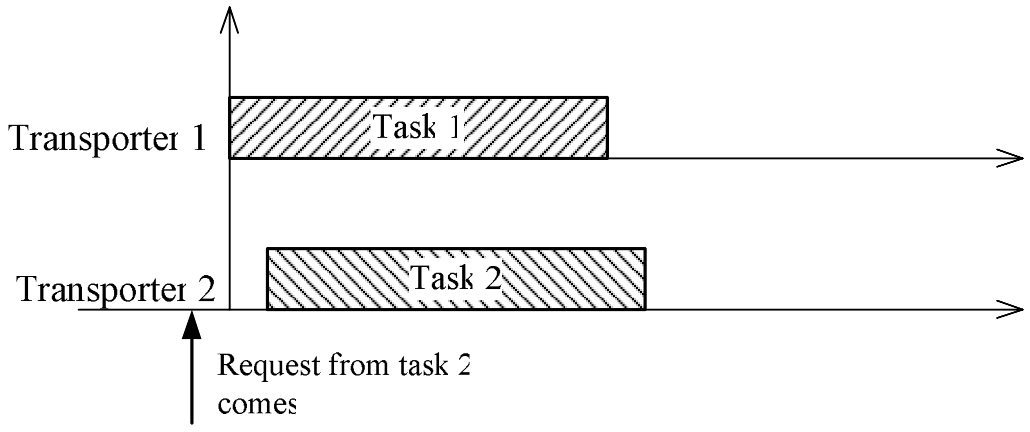

Figure 9.4: Stage 2 under stochastic distributed scheduling approach

3. With time *the request 3 from task 3 comes*. At this moment all transporters are busy, so it should wait till one of the transporters will become free (Figure 9.5).

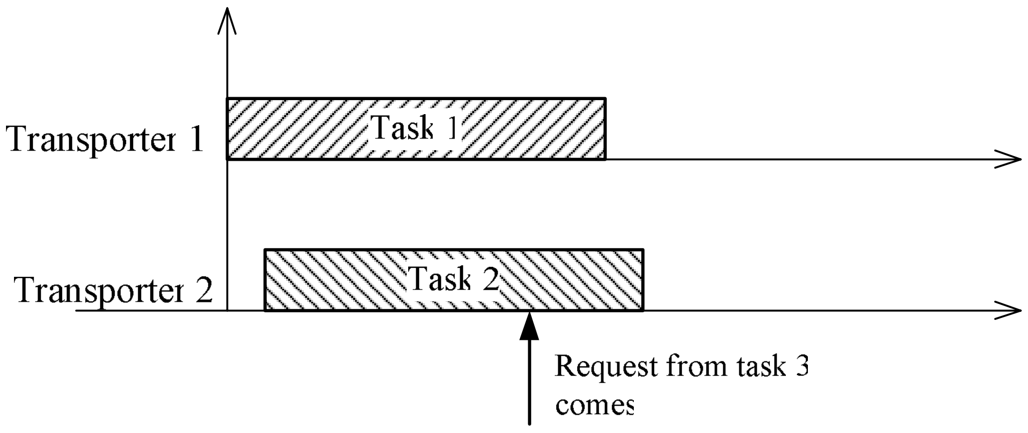

Figure 9.5: Stage 3 under stochastic distributed scheduling approach

4. When t*ransporter 1 is released,* task 3 will be assigned to transporter 1. Executing the task 3 this transporter brings utility 15 (Figure 9.6)

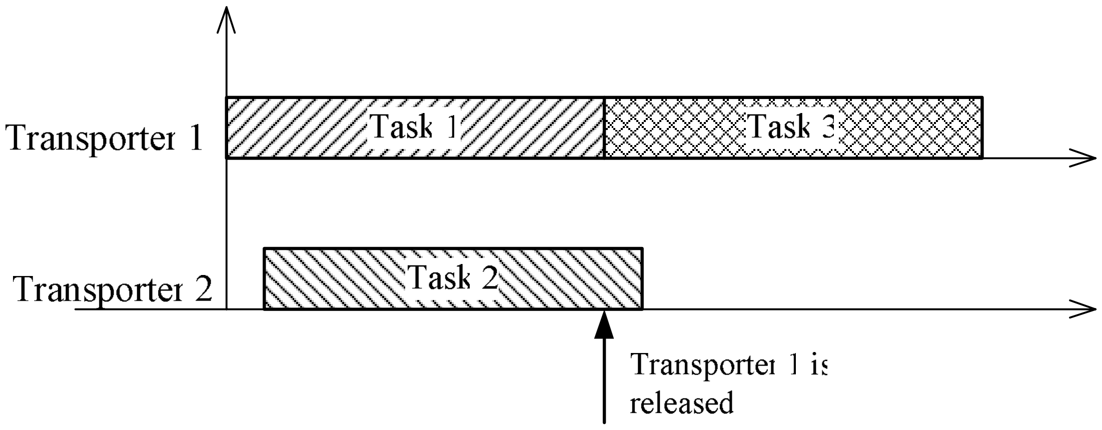

Figure 9.6: Stage 4 under stochastic distributed scheduling approach

It is worthwhile to mention, that the sum of utilities of all three tasks is *8 + 3 + 15 = 26.*

9.3 Stochastic Team Scheduling Approach

Under stochastic team scheduling the activity is planned for short-termed period of times, and the rescheduling when new tasks are coming or transporters are being released.

Thus, the coordination between tasks will consist of the following stages:

1. *The request 1 from task 1 comes*. Like under previous approach, the request is received by all transporters and all transporters send the utility of task 1 in answer. The workpiece selects transporter 1 because it brings the highest utility, namely utility 8 (Figure 9.7).

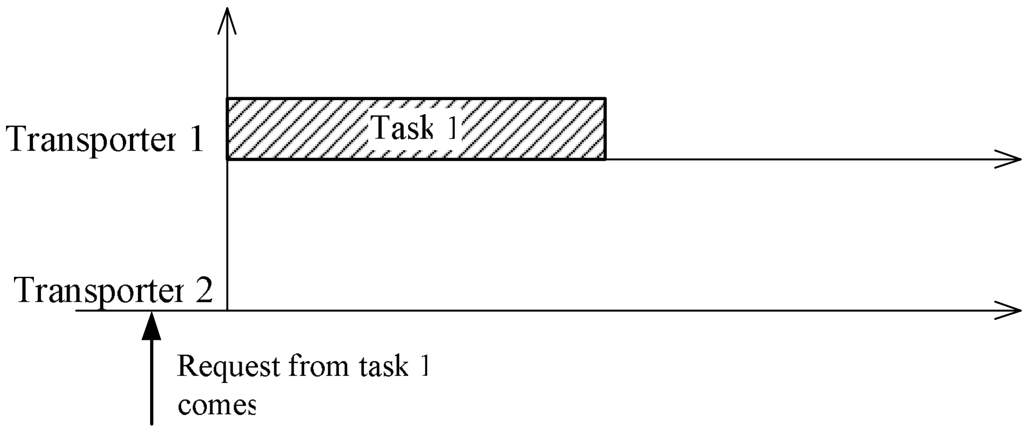

Figure 9.7: Stage 1 under stochastic team scheduling approach

2. *The request 2 from task 2 comes*. This request is received by all the transporters. Among both transporters transporter 1 can give the highest utility. Moreover, this utility is higher than the utility of executing task 1 by this transporter. In that case task 1 will be cancelled and transporter 1 will take task 2. After that task 1 will try to find a transporter that will take it. Obviously it will be transporter 2. So, after this stage task 1 will be assigned to transporter 2 (that brings utility 6) and task 2 will be assigned to transporter 1 (that brings utility 10) (Figure 9.8).

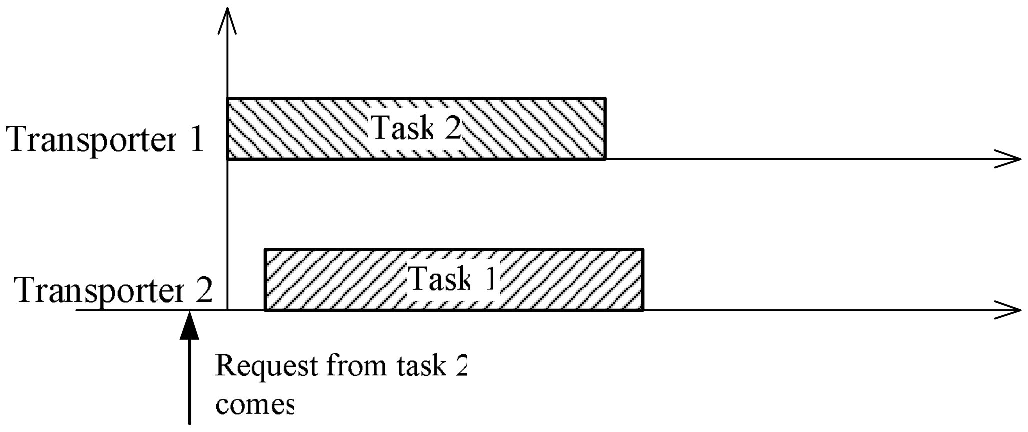

Figure 9.8: Stage 2 under stochastic team scheduling approach

3. With time *the request 3 from task 3 comes*. At this moment all transporters are busy, so it should wait till one of the transporters will become free (Figure 9.9).

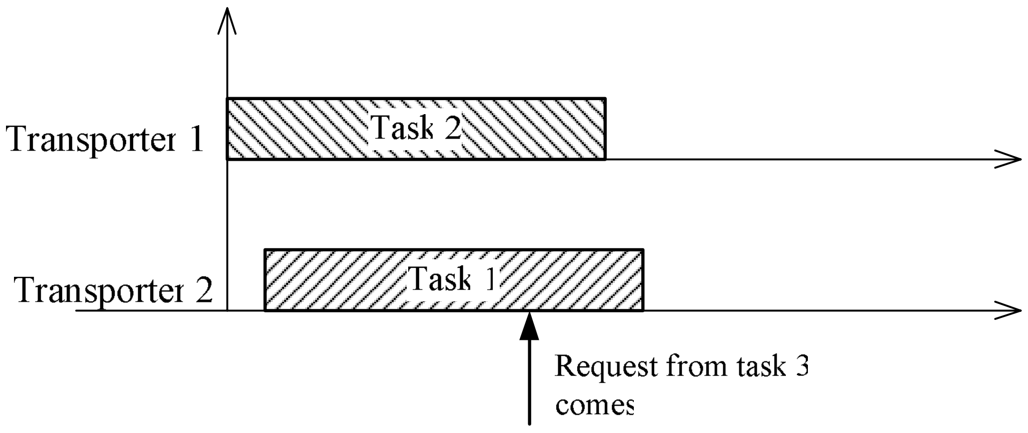

Figure 9.9: Stage 3 under stochastic team scheduling approach

4. When t*ransporter 1 is released*, task 3 will be assigned to transporter 1. Executing the task 3 this transporter brings utility 15 (Figure 9.10)

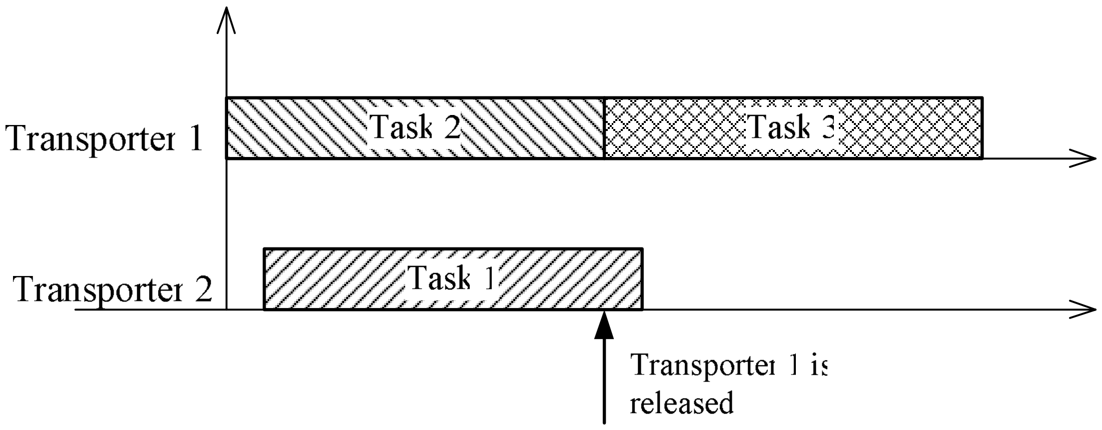

Figure 9.10: Stage 4 under stochastic team scheduling approach

It is worthwhile to mention, that the sum of utilities of all three tasks is *6 + 10 + 15 = 31.* It is clear that the utility is 6 units greater than under stochastic distributed scheduling. That is because stochastic team scheduling did optimization on the stage 2, and ascertained that it is better to transfer execution of task 1 to the transporter 2, and allocate task 2 to the transporter 1.

10 Summary and Outlook

10.1 Summary

In this project the problem of coordination and collaboration in multi-agent systems was analyzed and the coordination technique for flexible production environment automation system was developed.

First the definition and characteristics of a production environment, for which the coordination technique is elaborated, were given and analysed. Also the IAS model process “Lego-robots” that will be used as an example of a flexible production environment was briefly described. The second object of the work was the agent-oriented methodology explanation. For that the terms “agent” and “multi-agent system” were defined, the basic concepts of methodology were revised, classification of multi-agent systems was carried out and the Gaia methodology was explained.

The further important part of the work was concerned with coordination and collaboration problem within multi-agent systems. In this part the analysis of coordination techniques was carried out. It was explained in detail why agents need to coordinate and which properties in this connection an adequate coordination theory should have. Then in the literature known approaches to coordination in flexible production environment were analyzed and developed. In particular, four approaches were listed: hierarchical (centralized) approach, contract net protocol based approach, team scheduling approach and stochastic distributed scheduling approach. All this approaches in different ways resolve the problem of coordination in multi-agent automation system for flexible production environment.

After that an agent-oriented analysis of the system was carried out. As a result of it roles and interactions models in the flexible production environment automation system were defined. The roles model gives information on which roles are present in the system and which actions they carry out. The interactions model defines, which interactions occur between these roles. Next the activity selection problem was analyzed and activity selection mechanism was developed. It was defined, how agents can execute a global task, which is characterized by different parameters in a distributed way, in other words which is described in terms of different sequential goals.

As the key point of the project the coordination technique for the flexible production environment “Lego-robots” was developed. First it was analyzed, how successful different coordination techniques for coordination in flexible production systems may be applied in “Lego-robots” environment. After that the stochastic team scheduling approach was elaborated, which represents a combination of two approaches, namely the team scheduling and the stochastic distributed scheduling. The last point of the work was concerned with evaluation scenarios elaboration with the aim to give examples, how team scheduling, stochastic distributed scheduling and stochastic team scheduling coordination techniques work.

Thus, the characteristics of flexible production environment, the description of agent-oriented methodology concepts, analysis of coordination techniques and their application to given environment were examined. This served to the development of coordination mechanisms for flexible production environment on the example of the model process “Lego-robots”.

10.2 Experiences

First of all the content of the work was very interesting for me. Though known to me object-oriented methodology is allied to agent-oriented methodology, the agent-oriented methodology was still new for me and I studied it almost from scratch. It was a rather complicated experience but a very interesting and useful one. So, analysis and development of coordination techniques for multi-agent system indeed brought a lot of pleasure to me.

The project was carried out according to the requirements of the IAS Regulations Model. I find this model as a very comprehensive and useful thing that gives possibility to carry complicated projects in effective way, because of the division of the project into several sufficient parts with execution each of them according to the project plan.

The last but not the least is very positive impressions that the tutorship of Dipl-Ing. Hisham Mubarak left. Despite the intensive work and problems concerned with this infectivity of the project and fulfilment of project plan, he was always open to questions and ready to very helpful and interesting discussions.

10.3 Problems

During this voluntary undergraduate project no significant problems turned out.

Troublesome factor that brought a lot of difficulties was a short term of project course. Instead of 2,5 months it was only six weeks, so the work was quite intensive, and it was rather difficult to keep project plan.

Conceptual problems were mainly connected with the analysis of two known coordination approaches, namely team scheduling and stochastic distributed scheduling approaches, and development of new coordination technique for the “Lego-robots” environment based on these two approaches. The differences between these to approaches were not obvious, so it took a lot of time to clarify them and determine which coordination technique gives the best for to the flexible production environment.

10.4 Outlook

There are several points that need the revision that may be improved.

The first point is the dynamical assignment of the priorities to the different goals. The priorities are realized with the help of higher o lower coefficients in the weighed sum that combines partial utilities. At present time this coefficients are assigned statically, before the transportation process begins. However it is a good idea to make it possible to change them in run-time, by the user or by the system itself. That gives the possibility to change the priority of the defined tasks, for example set a higher priority of the transport optimisation partial utility for a robot that has some mechanical problems, or to assign a higher priority to the real-time requirements in case when the system is behind the schedule.

The second point is an organisation of a team formation mechanism. That is necessary because the proposed approach gives its best only in small domains. With the growth of robots number and territorial distribution of transportation tasks the efficiency of the proposed coordination technique may decrease. To avoid that, teams of robots may be formed depending on their features and territorial location. After that the described coordination processes should take place between robots of the same team.

Finally, the last point is of course the implementation of the proposed coordination technique for the flexible production environment model “Lego-robots”. For that a precise analysis and design of the system should be done, and then the implementation may be carried out.

Abbreviations

AI	**A**rtificial **I**ntelligence
CNP	**C**ontract **N**et **P**rotocol
DAI	**D**istributed **A**rtificial **I**ntelligence
MAS	**M**ulti-**A**gent **S**ystem
MaSE	The **M**ulti-**a**gent **S**ystems **E**ngineering methodology

Terminology

Agent	A delimitable software unit with defined goal. Being autonomous, an agent tries to achieve this goal and for that it interacts continuously with the environment and other agents.
Coordination	A process by which agents engage in order to ensure their community acts in a coherent and harmonious manner. In other words coordination make multi-agent system act like one unit.
Contracting Coordination	A coordination approach that is based on decomposition the problem into subproblems and attempts to find the agents with necessary resources to solve these subproblems.
Multi-Agent System	A system that consists of a group of agents that can potentially interact with each other
Multi-Agent Planning	A coordination approach by which the agents are engaged in multi-agent planning. In order to avoid inconsistent or conflicting actions and interactions and organize collaboration, agents build a multi-agent plan that details all their future actions and interactions required to achieve their goals, and carry out the execution of plans with more consequent planning and re-planning.
Negotiation	The communication process of a group of agents in order to reach a mutually accepted agreement on some matter.
Organizational Coordination	The simplest coordination approach which suppose the *a priori* organisational structure. On the other words, organisation defines implicitly the agent's responsibilities, capabilities, connectivity and control flow. It also defines roles, communication paths and authority relationships. Thus, with organisational coordination long-term relationships between agents are predefined.
Task Environment	The problems to which agent- and multi-agent systems are the solution.

Literature

[BuMu92] **Bussmann, S.; Muller, J.:** *A Negotiation Framework for Co-operating Agents*. In: Deen, S. M. (ed.), Proc. CKBS-SIG, Dake Centre, University of Keele. 1992, 1-17.

[EPR95] **Ephrati, E.; Pollack M.E., Rosenschein, J.S.:** *A Tractable Heuristic that Maximiyes Global Utility through Local Plan Combination*. 1995.

[GHN+97] **Green, S.; Hurst, L.; Nangle, B; Cunningham, P.; Somers, F.; Evans, R.:** *Software Agents: A review*. 1997.

[Göhn04] **Göhner, P.:** *Industrial Automation*. In: Lecture Notes. Summer term 2004.

[Fern03] **Fernandez, P.F.G.:** *Development of an Agent Oriented Control Application to a Railway Model*. Master Thesis. 2003.

[JSW96] **Jennings, N.R., Sycata, K., Woodridge, M.:** *A roadmap of agent research and development*. In: Int. Journal of Autonomous Agents and Multi-agent Systems. 1996, No. 1 (1), pp7…38.

[Haye95] **Hayes-Roth, B.:** *An architecture for adaptative intelligent systems*. In: Artificial Intelligence: Special Issue on Agents and Interactivity 72. 1995, pp329…365.

[HoRa97] **Holthaus, O.; Rajendran, C.:** *New Dispatching Rules for Scheduling in a Job Shop – An Experimental Study*. In: International Journal of Advanced Manufacturing Technology. 1997, 13, No.2, pp. 148…153

[KrMa91] **Kreifelt, T.; von Martial, F.:** *A negotiation framework for autonomous agents*. In: Y. Demazeau & J. P. Muller (eds), Decentralized A. I. 2, Elsevier Science. 1991.

[LRP02] **Lorenzo, C.; Raffaele, P.; Pierluigi, S.**: *Real Time Scheduling Using a Dispatching Rule Based Autonomous Agent System*. 2002

[Maes95] **Maes, P.:** *Artificial Life Meets Entertainment: Life like Autonomous Agents*. In: Communications of the ACM. 1995, No. 38, 11, pp. 108…114.

[Modi03] **Modi, P.J.:** *Distributed Constraint Optimization for Multiagent Systems*. 2003

[MuBo96] **Musliner, D.J.; Boddy, M.S.:** *Contract-Based Distributed Scheduling for Distributed Processung*. 1996.

[NLJ96] **Nwana, H.; Lee, L.; Jenings, N.:** *Coordination in Software Agents*. In: BT Technology Journal. 1996, 14(4).

[RaGe95] **Rao, A.;Georgeff, M.:** *BDI Agents: From Theory to Practice.* In: Proceedings of the First International - Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA. 1995.

[Resc04] **Resch, M.:** *Development of an Agent Oriented Controlled Production Process Model*. Diplomarbeit. 2004.

[RoZl94] **Rosenschein, J.S.; Zlotkin, G.:** *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. In: MIT Press. 1994.

[RuNo03] **Russell, S.; Norvig, P.:** *Artificial Intelligence: A Modern Approach*. Prentice Hall. 2003.

[SaFo89] **Sathi, A., Fox, M.S.:** *Constraint-directed negotiation of resource allocations*. In: L. Gasser & M. Huhns (eds), Distributed Artificial Intelligence *2*, Morgan Kaufmann. 1989.

[Tvei01] **Tveit, A.:** *A survey of Agent-Oriented Software Engineering*. In: First NTNU CSGSC. 2001.

[Vlas03] **Vlassis N.:** *Multiagent Systems and Distributed AI*. 2003.

[Weis99] **Weiss, G.:** *Multiagent Systems*. In: MIT Press. 1999.

[Werk90] **Werkman, K.J.,:** *Knowledge-based model of negotiation using shareable perspectives*. In: Proc. of the 10th Int. Workshop on DAI, Texas. 1990.

[WJK00] **Wooldridge, M.; Jennings, N.; Kinny, D.:** *The Gaia Methodology for Agent-Oriented Analysis and Design*. In: Autonomous Agents and Multi-Agent Systems. 2000, no. 3, pp.285…312.

[WoJe94] **Wooldridge, M., Jennings, N**.: *Agent theories, architectures and languages: A survey*. Intelligent Agents, Agent Theories, Architectures and Languages (ATAL 94), Vol. 890 Springer-Verlag Lecture Notes in Artificial Intelligence. 1994. pp1…32.

[WoJe99] **Wooldridge, M.; Jennings, N.:** *The Cooperative Problem-Solving Process. In: J. Logic Computat*. 1999, Vol. 9, No.4, pp.563-592.

[WUG03] **Wagner, T.; Göhner, P.; Urbano, P.:** *Softwareagenten – Einführung und Überblick über eine alternative Art der Softwareentwicklung.* In: atp - Automatisierungstechnische Praxis. 2003, pp. 3...31.

Index of Figures

Index of tables

Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und nur die angegebenen Quellen benutzt zu haben.

Unterschrift:

Stuttgart, den 26.01.2005