Chapter 4: Multi-Robot Architectures

Multi-robot architectures can be seen as a special case of multi-agent systems, where agents are embodied in their environment. This chapter focuses on multi-robot architectures specifically, since a more complete overview of MASs architectures is outside the scope of this thesis. Section 4.1 enumerates some of the early MASs together with a taxonomy for multi-robot teams. A behaviour based robotics approach to multi-robot teams is discussed in section 4.2. A hybrid multi-robot architecture, MACTA, is overviewed and discussed in section 4.3.

4.1 Introduction

In this section, some of the general MASs are overviewed. The first MAS architectures and related techniques have their origin in the Distributed Artificial Intelligence (DAI) field. The Contract Net Protocol (CNP) [49][175] can be considered as one of the first illustrations of the concept of agency. CNP is used to divide a task between multiple agents and it is a widely used protocol with many variations [161][46].

One of the first models to have actually implemented agents (albeit very simple agents) is ACTORS [1]. Since then many MASs have been implemented in fields as diverse as air traffic control [41], distributed vehicle monitoring [59] and more recently, Intrusion Detection Systems (IDS) [7]. As stated previously, the emphasis of this thesis is on embodied agents (robots) and from the next section, all classifications and examples are robotics-related. The reader is referred to [88][138] for more detail on general MASs.

With robotic MASs in mind, it is useful to consider a robotic MAS taxonomy. The chosen taxonomy for the purpose of this thesis is that of Dudek *et al* [56]. According to Dudek *et al* , robot teams are classified according to the following characteristics:

- Size of the team. Based on the size of robot teams, teams can be divided into classes ALONE, PAIR, LIM and INF, denoting one, two, multiple (but with a finite limited number of robots) and an unlimited number of robots respectively.
- Communication range. The teams are classified based on the communication range into NONE, NEAR and INF classes, denoting no communication, local communication (limited to the distance between robots) and unlimited distance communication (for example softbots, that can communicate using the Internet).
- Communication topology. According to the communication topology, multi-robot teams are classified as: BROAD, where a broadcasting mode of communication is used; ADD, where each agent has an address; TREE, where communication is done through a hierarchical network; and GRAPH, where communication links are defined as a graph.
- Communication bandwidth. Based on the cost of the communication, robot teams are divided into four classes: INF, where communication is free; MOTION, where the cost of communication is the same order of magnitude as the motion of the robot; LOW, where the cost of communication is greater than the cost of moving the robot; and ZERO, which indicates that no communication is possible.
- Collective reconfigurability. The collective reconfigurability indicates the rate at which the robot team can re-organise itself spatially. It divides the robot collective into three distinct classes: STATIC, where the topology is fixed; COMM, where members coordinate to achieve the reorganisation task; and DYN, where the spatial relationship can change arbitrarily.
- Processing ability of each collective unit. Robot teams are divided into classes based on the computational model of each robot. The view adopted in this thesis is that classification based on computational model is too limited, as it does not take in consideration the real processing power or complexities of the adopted world model representation (if any). However, for the sake of completeness, the four classes are enumerated next: SUM, where processing power is equivalent to that of a non-linear summation unit (i.e. a neuron in artificial neural networks); FSA, where processing power is equivalent to a finite state automaton; PDA, where processing power is equivalent to a push down automaton and TME, where processing power is that of a Turing machine equivalent. For more detail, the reader is referred to [56].

- Collective composition. A robot team can be classified according to the physical attributes of robots: IDENT, where all agents are identical; HOM, which indicates homogenous MAS where agents essentially have the same characteristics; and HET, where agents are heterogeneous with different physical characteristics. For a formal approach to measuring robot group diversity, reader is referred to [12][14].

In the remainder of this chapter, two multi-robot architectures, the Behaviour Based Robotic (BBR) and Multiple Automata for Complex Task Achievement (MACTA) are overviewed and discussed. Note that the BBR has evolved over time, but for the purpose of this thesis the first, original, version as described in [113] is considered.

Using the adopted taxonomy, these two architectures can be described as given in table 2.

Robot Team Characteristic	BBR	MACTA
Size of Team	LIM	PAIR
Communication Range	NEAR	NONE
Communication Topology	BROAD	Not Applicable
Communication Bandwidth	MOTION	Not Apllicable
Collective Reconfigurability	DYN	STATIC
Processing Power of a Team Member	TME	TME
Collective Composition	HET/HOM	HET/HOM

Table 2 Comparison of BBR and MACTA architectures

4.2 Behaviour Based Robotics

There were various efforts to improve on reactive architectures capabilities. Some of these effort were mentioned in section 3.3.3. The behaviour based robotics architecture, as developed by an MIT team headed by Matarić [109], is discussed to illustrate the ideas behind a behaviour based approach for MAS.

4.2.1 Introduction

Behaviour based robotics can be seen as an extension of purely reactive architectures [108]. Behaviour based robotics has the concept "behaviour" as its foundation. In addition to the definition of behaviour as given in section 3.4.2.2, a behaviour can also be defined as a piece of code that produces behaviour when it executes [74].

BBR provides mechanisms for cooperation, coordination, communication and planning [109]. In general, BBR are decentralised systems of autonomous agents. Cao *et al* [38] consider BBR to be a swarm-like architecture. This view can be accepted to a degree, but BBR is certainly more advanced than a classical swarm system such as ANT [119N, 181] as it allows for explicit communication and learning.

BBR adopts a building block approach, similar to that used by the subsumption architecture (see section 3.3.3), that relies on developing basic behaviours first. Once basic behaviours are thoroughly tested, the basic behaviours are then combined into more complex behaviours.

Behaviour based agents can become quite complex when basic behaviours are combined in more complex ways, or when the side effects of its behaviour and interaction with the environment can yield some useful characteristics, such as emerging behaviour [178].

The concept of intelligent emerging behaviour is a paramount premise of reactive and behaviour based architectures. The belief that intelligent behaviour will emerge through the interaction of a system with its environment is based on the observation of natural systems, for example ant colonies [178]. However, there is still no formalisation of a process that will allow new intelligent behaviours to emerge in multi-robot teams. In this thesis, proposed emergent behaviour is exhibited on a higher level of abstraction – the social level, thus the topic of emergent low-level behaviours through interaction with an environment falls outside the scope of this thesis.

BBR uses a decentralised approach where all robots are fully autonomous with sparse communication between robots. Furthermore, knowledge representation is not symbolic. Hence there is no high level planner nor any symbolic learning mechanism.

4.2.2 Basic Behaviours

The building block approach of BBR assumes that simplistic basic behaviours can be combined into more complex ones at a higher level of abstraction. BBR also proposes a methodology for choosing such basic behaviours. That being said, the process of choosing basic behaviours is still a heuristic process, without a fixed metric for selecting an "optimal" set of basic behaviours. Matarić proposes two criteria for defining the set of basic behaviours [113]:

- *Necessity*. The set of basic behaviours must contain only behaviours that are necessary to achieve the desired goals in a given domain.
- *Sufficiency*. The set of basic behaviours must contain behaviours that will be sufficient to achieve the desired goal in a given domain.

Benchmark tasks for robot teams that must be achieved by combining basic behaviours are foraging, flocking, herding and surrounding. To achieve these tasks (assuming a two-dimensional spatial domain) the following basic behaviours have been identified [110][113]:

- *Safe wandering*, which refers to the ability of a robot team to move around while avoiding obstacles and collisions with other agents. This behaviour is the basis for any exploration or mapping task.
- *Following*, which refers the ability of an agent to follow another agent (the leader) and maintaining a following distance from the leader.
- *Dispersion*, which refers to the ability to spread out and to maintain a minimum distance between agents.
- *Aggregation*, which is the opposite of dispersion, i.e. the ability to gather and maintain a maximum distance between the agents.

- Homing, which refers to the ability to find a particular region or location. This ability is necessary for tasks such as sample collection, foraging and self-preservation.

BBR has been designed as an architecture that can be easily extended by implementing more complex behaviours. This characteristic is desired and required for an emergent behaviour approach.

The strength of a behaviour based architecture also lies in the fact that incorporation into more complex hybrid architectures is easy without any significant modifications. The behaviour based architecture is used in some well known hybrid MAS such as MACTA [11][10] and InterRap [129].

4.2.3 Learning in BBR

The goal of any learning mechanism is to optimise system performance. There are various applications of learning mechanisms that can improve the performance of the system [91]. Applied to BBR, learning can be divided into learning new behaviours, learning new facts about the environment and learning to improve behaviours. It is important to note that learning in BBR is not based on symbolic reasoning. In BBR, learning uses behaviours that are in their nature sub-symbolic structures.

There has been significant improvement in the learning capabilities of BBR MASs over the past few years. Initially, research focused on learning behaviour policies, in other words on improving the behaviour selection process through learning [113]. More recent research has complemented the behaviour policies learning with learning from environment models and from interaction with other agents [115]. An overview of learning in BBR is given next.

4.2.3.1 Learning Behaviour Policies

The behaviour policies determine which behaviours are selected for execution. The goal of learning behaviour policies is to improve on the selection of appropriate

behaviours for specific tasks and environment conditions. Reinforcement learning [91] is one of the most frequently used mechanisms in AI. In BBR, the reinforcement learning is used to learn behaviour policies. Reinforcement learning relies heavily on stimuli-response coupling, and it has been used in robotics, for example in [25].

Most reinforcement learning models that were successfully applied to computational learning are based on Markov Decision Process (MDP) models [113]. Unfortunately MDPs, when applied to embodied, situated agents, assumes that interaction between an agent and its environment can be viewed as synchronised finite-state automata that are deterministic and predictable. This is not always the case, due to uncertainty in sensing of the environment. For MDP to be applicable to learning of behaviour policies, the following modifications have been made [113]. These modifications are described next.

- In robotic applications of MDP, the state space defines sensor inputs of the robot, together with the set of internal parameters of the robot. The first modification is that the state space has been defined by conditions instead of full state descriptions. The space of conditions is generally much smaller than the complete state space and allows for faster computations.

- The reward function is focused on achieving pre-determined sub-goals of concurrent behaviour. This is a very important departure from the original MDP model that prefers a sequential process. At any moment, in BBR robotic applications there are many active behaviours, contributing by their actions towards a completion of a high-level goal. Concurrent reward functions can then focus on improving each behaviour. The improvement in behaviours leads to the improvement of overall system performance.

- A progress estimator function is introduced as a part of the learning mechanism. The role of the progress estimator function is dual: instead of rewarding only after a specific goal has been achieved, an intermittent reward allocation is used. The progress estimator function uses a mechanism for a termination of a specific behaviour [113].

The MDP was further modified, by using reward sharing mechanism that punishes greedy behaviour of agents [116][118].

4.2.3.2 Learning Environment Model

One of the main characteristics of BBR is a decentralised approach and a lack of symbolic representation of its environment. Modelling of the environment in BBR is reduced to creating a world map, based on exploration. Even learning of such a simplified environmental model presents a challenge to BBR architectures. The problem of learning a world map was solved through the mapping based on location of landmarks [112]. In BBR there is no provision for storing traditional symbolic knowledge that can describe a world map. Instead world map information is stored by creating behaviours that store landmark information, as described next.

The mapping based on location of landmarks proposes that once a landmark is detected, a new behaviour is created. The newly created behaviour has the following parameters: landmark type, direction (orientation) and distance from the previous landmark. Each new behaviour is linked to the previous one, in effect creating a map that consists of a set of linked landmarks. Navigation then consists of traversing from one landmark to another.

More details about the landmark navigation approach to mapping can be found in [112][50].

4.2.3.3 Learning Behaviour Patterns from Behaviour History

The BBR architecture incorporates a mechanism to learn behaviour patterns from the history of behaviour dynamics [124]. Each robot maintains a tree-like topological structure for the duration of task execution, wherein occurred behaviours are stored.

The tree topology represents the robot's behaviour space, where each node represents an executed behaviour. The tree topology is the representation of a behaviour

sequence in a robot's behaviour space. The links were weighted statistically, according to link-usage frequency.

For the different goals, patterns in behaviour activations are identified using the tree of behaviours. If a pattern in behaviours activation has led to the fulfilment of the identified goal, the identified pattern is used in subsequent behaviour selection of similar tasks. The approach was successfully tested on robot systems. More details on these experiments can be found in [124].

4.2.3.4 Other Learning Methods in BBR

Various other learning methods were tested in the BBR architecture, some using innovative approaches. For example, to facilitate learning of interaction models between agents, the authors have developed Augmented Makarov Models (AMM) [119]. In order to provide a higher level of abstraction, necessary for more complex deliberation, an abstract behaviour was introduced in [135] that have provided BBR with reasoning tools almost equal to those of the best deliberative, symbolic systems.

4.2.4 Cooperation Model

Cooperation does exist in BBR but it is not obvious, as there is no intentional cooperation. Instead, cooperative behaviour can emerge as a side-effect. At this stage it is important to note a view firstly expressed by Matarić [114] that cooperative behaviour based on negotiations require direct communication between the agents. Communication is also a prerequisite for distributed cooperative problem solving [80]. With these views in mind, the cooperation model used in BBR has limited capabilities, mostly because of the limitations of the communication method.

Firstly, BBR is a highly decentralised architecture with fully autonomous agents. There is no central system or knowledge repository that can store the data that was acquired by multiple agents. Instead, each agent is dependant on its local environment for the knowledge it acquires. In other words, the knowledge is not shared.

Secondly, the communication method is capable of broadcasting only short messages. This further limits the amount of information that can be transferred between the agents (note that the communication protocol transmits only simple messages, not knowledge).

Thirdly, there is no symbolic knowledge information. The symbolic knowledge is easy to exchange and modify, using some of the more advanced approaches such as KQML [66][99] or the XML [186].

Initially, communication was used for the simple task of detecting another robot [113]. At a later stage, a robot could "agree" on team interaction based on information received from another agent [119]. Limited social behaviour was modelled using the same, limited communication mechanisms [117]. Despite the above-mentioned limitations, cooperation was investigated in [174] with a limited success.

Social behaviour is a way of resolving conflicts that arise in teams of autonomous, uncoordinated robots. Conflicts in BBR do not arise because of the competitive nature of agents but due to the unintentional interference between the agents. Agents do not have conflicting goals but the interference may lead to negative interaction.

Out of the three coordination mechanisms that were discussed in section 2.4.1.1, the BBR architecture was modified to use social coordination. Social coordination is the predominant approach to coordination in BBR [117][116].

4.2.5 BBR - Conclusion

As a successor to purely reactive architectures, BBR has continued the tradition of revolutionising the field of mobile embodied agents.

BBR has retained all positive characteristics of reactive architectures. BBR is characterised by fast execution time, due to simplistic couplings between sensors and actuators. BBR also lends itself to relatively simple robot implementations that provide an efficient environment for multiple robot team experiments.

BBR has significantly improved over purely reactive architectures by providing mechanisms for the internal representation of the environment [50], learning and improving on existing behaviours [118], communication [119] and even social behaviours [117].

One of the main characteristics of BBR is its limited communication model, which may seem to be a major limitation. However, considering the limited knowledge that is maintained by each agent, the limitations of its communication model are not so severe.

Cooperation, internal representation of the environment and even social behaviours are implemented in a very efficient manner, but all of them have limitations, as described in previous sections.

It is interesting to note that BBR has evolved from reactive architecture characteristics toward characteristics that are traditionally associated with symbolic architectures (e.g. planning, social interaction and environment modelling).

One of the main reasons for the appearance of reactive architectures and subsequent improvements of the reactive architecture (such as BBR) was the fact that the execution of traditional symbolic reasoning based systems was slow. Today, processing power is easily and cheaply available and this approach should be reconsidered. It seems that the future of behaviour based systems lies in its incorporation with hybrid systems as an environment-agent layer.

4.3 Hybrid MAS (MACTA)

This section presents an overview of a hybrid architecture, Multiple Automata for Complex Task Achievement (MACTA), developed at the University of Salford [11][10].

4.3.1 Introduction

MACTA combines a symbolic planner with a behavioural architecture, the Behavioural System Architecture (BSA) [17][15]. MACTA is referred to as a MAS with a reflective agent that supervises multiple behaviour architecture agents (robots) [11]. MACTA is not only a hybrid in the sense that it combines sub-symbolic and symbolic reasoning, but it also consists of embedded agents (robots) and a non-embedded agent (reflective agent). MACTA has been successfully tested on tasks such as docking, cooperative relocation of objects and tracking using two robots. MACTA consists of two main components, namely BSA and the reflective agent.

Although it consists of two main components, MACTA is a three-layer hybrid architecture because it also contains an interface layer, referred to as the mission organiser.

4.3.2 Behavioural Synthesis Architecture

BSA [17][15] extends and improves earlier reactive architectures such as the subsumption architecture [31]. BSA allows for single and multiple robot systems. Within BSA, two types of robotic behaviour are identified:

- Egoistic, or self-interested behaviour where a robot pursues only its goals, without taking into consideration the goal of the whole multi-robot team.
- Altruistic, where a robot might not pursue its optimal state, in order to allow the whole multi-robot team to achieve its optimal state.

Egoistic (or self interested) behaviour is a typical characteristic in a single robot system. Altruistic behaviour is often desired in case of multi-robot systems.

BSA defines four layers, namely the self, the environment, the species and the universe layers (refer to figure 9):

- The self-layer contains strategies concerned with internal resources (e.g. battery levels).
- The environment layer contains behaviours that relate to the agent's interaction with the environment (such as obstacle avoidance).
- The species layer contains strategies for the interaction between agents (such as cooperation).
- The last layer, the universe layer, contains strategies for achieving tasks (such as object reallocation).

BSA agents have no representation of the world. Instead, the BSA agents are aware only of their local environment. BSA, in the tradition of many reactive and behavioural architectures, is horizontally layered, where each layer can receive inputs from the environment and each layer can produce actuator commands.

Where BSA departs from reactive architectures is that the actuator command is synthesised from all actuator commands and only the resulting command is sent to the physical actuator. This process is illustrated in figure 9. The actuator commands produced from a single behaviour are represented in a form of vectors in two-dimensional space. The synthesised output, represented as a vector on the right hand side of figure 9, is then the result of summing all individual vectors.

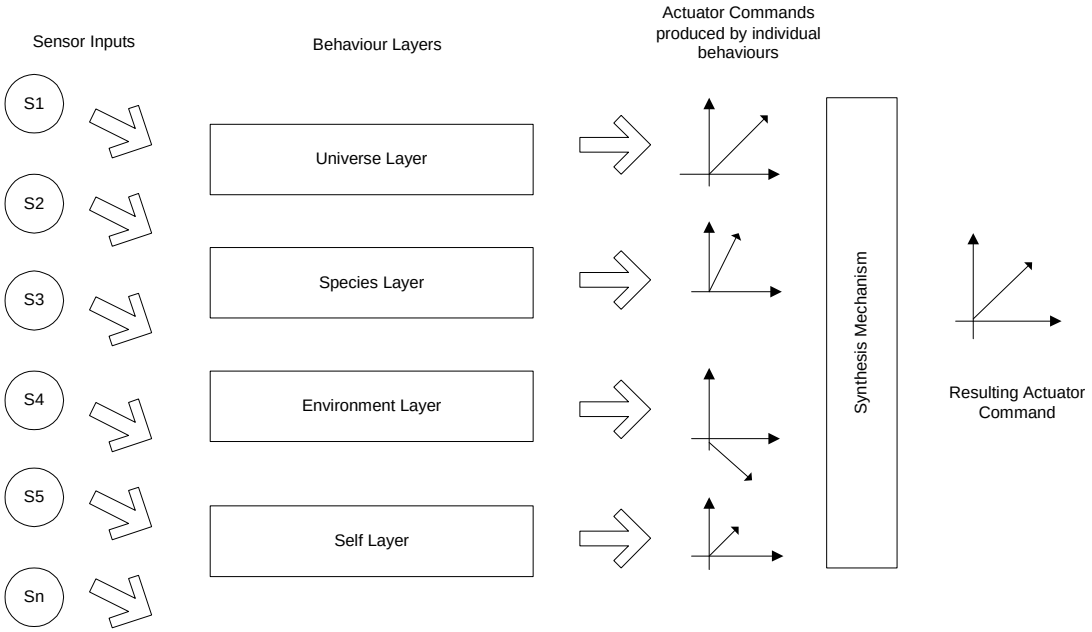

Figure 9. BSA Architecture illustrated (modified from [16]).

The synthesis mechanism is not the only coordination mechanism available to BSA. For example, coordination can be achieved by creating complex behaviours scripts with predetermined coordination rules. The complex behaviours can enable and disable simpler behaviours according to these coordination rules, thus preventing conflicting interaction between the behaviours.

4.3.3 Behaviour Scripts

Behaviour scripts serve as an interface between the behavioural architecture and the reflective agent. Behaviour scripts in BSA are in the form of a triplet (*sensor-pre-conditions, set of enabled behaviours, sensor post-conditions*).

The behaviour script can be seen as a set of enabled behaviours that are activated when sensor pre-condition is satisfied and remains active until sensor post-conditions are satisfied. It is important to note that scripts do not contain behaviours. The behaviours are contained in the BSA and organised in four layers, as explained in the previous section. The behaviours are activated or deactivated by appropriately setting the *active flag* associated with every behaviour, irrespective of the layer in which it is stored. The actuator output is then synthesised from the outputs of all active behaviours.

The process is illustrated below in Algorithm 1 (from [9]):

```
If (sensor-pre-conditions TRUE) then

        While NOT (sensor-post-condition)
                Synthesise(activebehaviours)
        Endwhile

endif
```

Algorithm 1. BSA behavior algorithm

Behaviour scripts correspond to the sequencer layer of hybrid architectures (see section 3.4.2.3). The original MACTA implementation used handcrafted behaviour scripts [11][16].

4.3.4 Reflective Agent

MACTA defines a fifth layer, the reflective agent, to maintain the symbolic world model and to perform symbolic reasoning. Although, logically, the reflective agent forms another layer of the MACTA architecture, the MACTA terminology refers to it as a reflective agent. In order to avoid possible confusion, the MACTA terminology is used from now onwards. The reflective agent has two main components: the mission organiser and the planner, both of which are described in this section. The planner creates a partial order plan (consisting of numerous sub-goals), using the symbolic reasoning mechanism, which is then passed to the mission organiser. The mission organiser then translates the symbolic sub-goals into behaviour scripts that are, in turn, passed to the agents (robots). A high level overview of the reflective agent is illustrated in a figure 10.

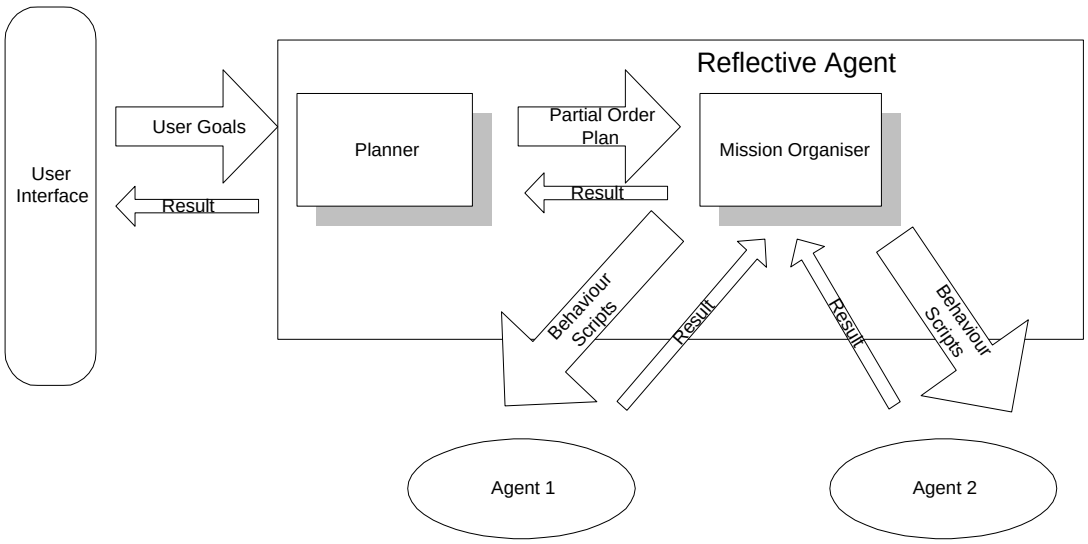

Figure 10. Overview of MACTA Reflective Agent

Interaction of a reflective agent with BSA agents is via behaviour scripts. The feedback from BSA agents to a reflective agent is very crude: the feedback is reduced to an indication as to whether the execution of a task has been successful or not. The planner and mission organiser are discussed next.

4.3.4.1 Planner

MACTA uses a standard symbolic planner [9], namely a modified UCPOP [146]. The UCPOP is modified as follows:

- Provision is made for multi-robot teams. Actions are, however, manually handcrafted.

- Task allocation is done in a simple way that uses all the available robots for a task. While this approach worked well for the experiments using the MACTA architecture (where only two robots have been used), it is unclear how (and if) this approach can be scaled up to robot teams consisting of a larger number of robots. An alternative task allocation mechanism using a market inspired approach, such as CNP [49], was considered but not implemented.

- The planner does not maintain a full world model. It contains only pre-programmed information about stationary objects, for example, walls.

The planner subdivides goals into a partially ordered plan that cannot be decomposed further. The partially ordered plan is then presented to the mission organiser.

4.3.4.2 Mission Organiser

The mission organiser's responsibility is to translate the symbolic representation of the user's desired goal (obtained via an user interface) and pass it to the planner. The planner then generates a hierarchical non-linear plan. The mission organiser translates the primitive (but still represented in symbolic terms) actions of a plan into behaviour scripts (that are represented in sub-symbolic terms), which are then propagated to appropriate agents. The role of the mission organiser is illustrated in figure 11.

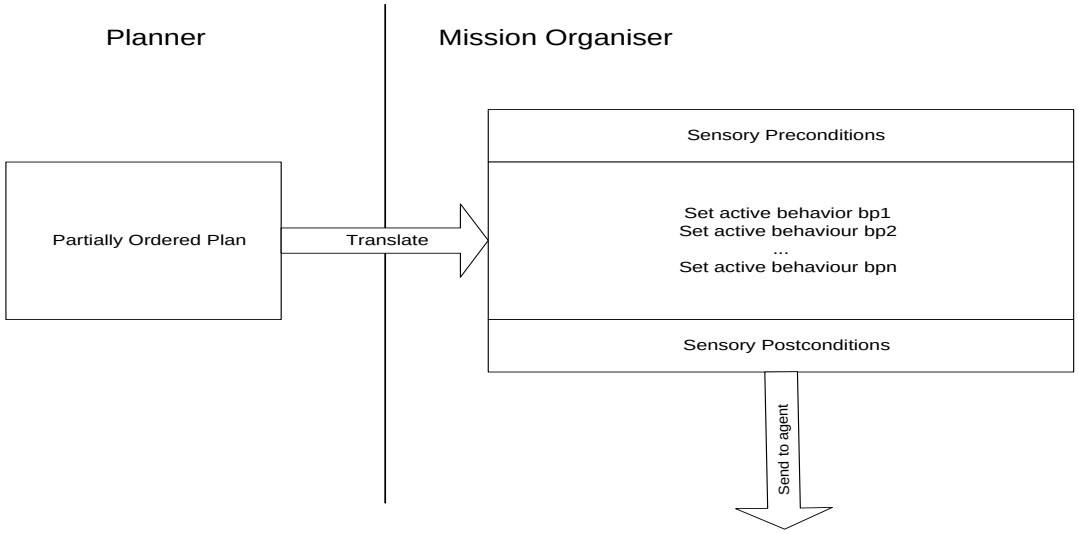

Figure 11. Role of the Mission Organiser in MACTA (adapted from [10])

4.3.5 Coordination Model

MACTA caters for coordination between agents in a very elementary fashion. Coordination is achieved through behaviours only, without exchanging any information and without maintaining a world model. In the experiments that were performed, MACTA was successfully applied to tasks such as cooperative object relocation and cooperative object tracking [11][10]. For each task, the reflective agent initiated coordination.

There are two modes of interaction between robots: interaction between close-coupled robots and interaction between loose-coupled robots. In MACTA, a close-coupled scenario assumes physical coupling, i.e. robots are physically attached to each other. In loose-coupling, robots are reliant on remote sensing equipment such as cameras.

The coordination model used in MACTA has numerous shortcomings. Firstly, the reflective agent is always the initiator of coordination behaviour. This is not ideal in a highly distributed environment where agents are complex because resource usage is inefficient. Secondly, all behaviours for coordination are handcrafted. It is improbable that handcrafted solutions can be scaled to large robotic teams. Thirdly, interaction between robots is done purely through sensing of the local environment. This is a characteristic of reactive and behaviour based architectures and a critique of these was presented in section 3.3.4.

In MACTA, although agents are autonomous, the coordination model used indicates that all planning is done centrally. MACTA can therefore be seen as a centralised architecture.

4.3.6 MACTA - Conclusion

The MACTA architecture is a promising hybrid architecture that combines the best characteristics of a symbolic planner with an efficient behaviour architecture. However, there are some aspects that need improvement. For example, all behaviour scripts are handcrafted, which is clearly not feasible for a large number of heterogeneous agents. The mission organiser component of the system can also be improved. The current system uses one-to-one mapping of a primitive action (the output of a planner component) to a behaviour script. Intuitively, an one-to-many mapping would potentially be more powerful (and abstract), but it would introduce additional complexity. In other words, a desired task could be achieved through different behaviours, depending on the conditions and inputs from other agents or the environment.

MACTA does not provide any learning mechanisms, which is another serious shortcoming. However, the primary problem with MACTA is that the majority of the model is either handcrafted or it caters for a team of only two robots, without a clear indication of how it could scale to larger teams of robots.

4.4 Summary

Two MAS architectures, together with a representative example, were overviewed and discussed in this chapter. Particular consideration was given to the coordination model employed by both architectures, as the coordination model is seen as the key to successful MASs.

The next chapter presents a new MAS architecture, INDABA, which addresses one of the most important aspects of MAS architectures, namely the ease of implementation of the coordination mechanism.