# Evaluating the Effectiveness of Multi-Agent Organisational Paradigms in a Real-Time Strategy Environment

Engineering Multiagent Systems Track

Buster A. Bernstein
Delft University of Technology
Delft, The Netherlands
b.a.bernstein@student.tudelft.nl

Jasper C.M. Geurtz
Delft University of Technology
Delft, The Netherlands
j.c.m.geurtz@student.tudelft.nl

Vincent J. Koeman
Delft University of Technology
Delft, The Netherlands
v.j.koeman@tudelft.nl

## ABSTRACT

We study the impact of using different organisational paradigms on the design and implementation of a Multi-Agent System (MAS) for Real-Time Strategy (RTS) games. We examine systems designed and implemented according to a specific paradigm on their performance in a practical scenario, as well as examining software-engineering concepts like size and complexity entailed by the according implementations. In contrast to related theoretical work, we deal with the practical constraints and implications of the paradigms by targeting the prototypical RTS game StarCraft: Brood War. Through careful analysis of this environment, agent systems for four separate paradigms that operate at different levels of autonomy and communication are designed, implemented, and evaluated by thousands of instrumented runs. One of the main findings is that using a central processing agent, e.g. in a market-based approach, increases task performance, but at the cost of increased code complexity.

## KEYWORDS

MAS; RTS; AI; StarCraft; swarms; hierarchies; markets

**ACM Reference Format:**
Buster A. Bernstein, Jasper C.M. Geurtz, and Vincent J. Koeman. 2019. Evaluating the Effectiveness of Multi-Agent Organisational Paradigms in a Real-Time Strategy Environment. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

## 1 INTRODUCTION

Real-Time Strategy (RTS) games continue to be a challenging domain for AI, mainly because of the huge search space and the uncertainty about what the opponent is doing (i.e., "fog of war") [22]. Multi-Agent Systems (MAS) can provide a layer of abstraction for humans to design intelligent behaviours and systems that can attempt to achieve their goal(s) in this domain [7]. Many different models and ways to develop (i.e., design and implement) a MAS have been proposed [8]. The current literature on models for logic-based AI contains various such models that are mainly compared on a theoretical basis. The implications of developing agent systems based on such models for complex environments that involve 'real-world effects' like latency, uncertainty and randomness remains largely unstudied [20].

*Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.* 

This paper describes the design and execution of an experiment using an RTS environment as a platform for a targeting-scenario in order to compare multi-agent organisational paradigms. We focus on the various ways in which a multi-agent system can be organised. We aim to evaluate the practical implications of developing a multi-agent system in a specific organisational paradigm. To this end, a sufficiently complex environment is needed where implementations of such models can be developed, executed and tested. The RTS game StarCraft BroodWar (SC:BW) has emerged as a unified testbed for multi-agent research [13, 21] and AI research in general [21, 26].

After evaluating related work in section 2, the experiment is designed in section 3 through both considerations in the SC:BW environment and design of the targeting-scenario. Next, by setting up comparable metrics on both a performance level and a software level in section 4, viable theoretical models are selected that are suitable for this scenario based on two criteria in section 5.

The separation criterion for these paradigms is based on the level of autonomy of the agents versus the level of communication between those agents. In order to diversify the approaches on autonomy versus communication, four different MAS paradigms from literature have been selected: individual, swarm, market and hierarchical. Each selected paradigm is carefully worked out a practical implementation for the SC:BW environment.

In section 6, the outcomes and differences in performance (i.e., effectiveness in win rate) between the organisational paradigms are reported, as well as the differences from a software engineering perspective as entailed by the obtained software metrics, like code size and complexity. The paper ends with a discussion of the implications of these results and suggestions for future work in section 7.

## 2 RELATED WORK

RTS games are widely regarded as an ideal testbed for AI [16, 21]. A prototypical RTS game like StarCraft involves long-term high-level planning and decision making, but also short-term control and decision-making with individual units. These factors and their real-time constraints with hidden information make RTS games like StarCraft ideal for iterative advancement in addressing fundamental AI challenges [23]. Multi-agent systems seem to be a good fit for addressing these challenges, allowing individual agents to reason about their tactical decision making whilst communicating to make decisions at a joint strategical level [13].

The work of Weber et al. [27] recognises the value of an agent-oriented approach to RTS AI. Their "EISBot" for StarCraft uses a reactive planner together with several external components like

case-based reasoning and machine learning. Similar to multi-agent systems, the concepts of percepts and actions are used. However, there is only a single 'agent' that is compartmentalised into several specific managers based on different tasks in the game. This approach is thus fundamentally a single-agent approach, whilst in this work, we instead aim to compare organisational paradigms for multi-agent systems.

Similarly, the work of Luotsinen et al. [18] compares twelve paradigms of agency in a virtual environment with elements of turn-based strategy games. However, their focus lies more on the areas of economy and growth, whereas this paper focuses on combat in an environment without turns. Moreover, the use of genetic algorithms and learning introduces great measures of uncertainty. In this work, in contrast, we limit randomness and uncertainty with regards to the methodology as much as possible by creating a consistent strategy across all paradigms and reducing external influences of the environment.

Corkill et al. [4] also evaluate the impact of agent organisations on task performance, but does so in different domains (i.e., more controlled simulations), varies on different aspects, and does not evaluate the software engineering aspects as we do here.

# 3 EXPERIMENT

In order to be able to identify differences between organisational paradigms without requiring a full-fledged StarCraft AI implementation, a specific scenario is needed that allows the multi-agent systems to be relatively straightforward yet complex and varying enough to be able to compare the impact of using different organisational paradigms. To this end, a targeting game in SC:BW is employed in this work. In other words, two armies that combat each other on a specific location on the map. Both of these armies consist of units with a certain amount of health points. Through attacking, the units can reduce the health points of units on the opposing army. When units have no more health, they can no longer attack and are removed from the battlefield.

The implementation of a multi-agent organisational paradigm controls one army, with the goal of defeating the opposing army. A main reason for choosing this specific scenario is that it does not entail factors such as building, scouting and economy. By removing these elements from the equation, there are less factors that can influence the outcome, and the degree to which the outcome is dependent on the organisational models is thus increased.

## 3.1 StarCraft: BroodWar

To measure the performance of each model accurately, the RTS environment should provide detailed control. That means when interacting with the environment, it is possible at any stage to inspect its state and to send commands to the environment to control the actions of the units that exist within it. StarCraft (Brood War) can be controlled using the Brood War Application Programming Interface (BWAPI[1]). In the context of this paper, SC:BW has two players: the player and the enemy player. All units belonging to the player can be observed and controlled through BWAPI. the control over the opposing army is given to the *built-in AI* of SC:BW, which is the default AI that is included with the game. Internally, the game executes and progresses on a frame-by-frame basis, emulating the concept of time[2].

*3.1.1 Repeatability.* At the start of a regular game of SC:BW, a few basic units of the different players spawn out of each others vision, and thus exploration and the building of military units is required. This lengthy process adds a lot of unwanted complexity. In order to make our experiment (quickly) repeatable and to allow fair comparisons between repeated executions, for each run, all units must have the same starting health and starting positions. The map builder *StarEdit* (which is provided by SC:BW) can be used to create a custom map with an altered initial configuration. A custom map thus allows each trial (i.e., running a battle from start to finish) to start in a constant and replicable state, from which the paradigms can be reliably compared.

## 3.2 Scenario Implementation

Even though the initial state of the scenario can thus be selectively pre-defined, there are still a number of factors outside of the control of an agent system that might significantly influence the result of a trial. Based on careful examination of the environment, a list of considerations is devised to capture the influential factors of which the effects can be prevented. Below, for each consideration, its undesirable effect is discussed together with a solution.

*3.2.1 Battlefield.* The entire battle must be held on a flat surface. *SC:BW has an internal chance that a shot misses, and this chance increases from* $\frac{1}{256}$ *to* $\frac{136}{256}$ *when shooting up-hill*[3]. The map thus uses the default ground level terrain in order to allow all units equal hit chances.

*3.2.2 Equal armies.* The two armies must be of equal size and strength. If the size or strength is different, for example by using units that do different amounts of damage, the chance that the outcome is decided by such a difference instead of the decisions made by the agent system vastly increases.

*3.2.3 Stationary units.* Units must be able to stay stationary. If they have to move, units can collide with each other because of their pathing, which causes uncertain behaviour[4]. Collision handling is an internal non-deterministic process in the SC:BW game engine; taking this into account would add an additional layer of complexity without being beneficial to comparing the agent systems. A solution to this problem is to use ranged units, as they can target different units without moving, and thus avoid this issue.

*3.2.4 Ranged combat units.* The implemented scenario specifically uses *Marines*[5], as these units satisfy the previous requirement and have been used before in research that also employs a SC:BW combat situation [3, 9].

*3.2.5 Unit placement.* Following from the use of ranged units, a related constraint is that all such units must be able to target each other, which is satisfied by placing them on the map within each others shooting ranges. The units are placed in a grid formation, as illustrated in Figure 1.

[1] https://github.com/bwapi/bwapi

[2] http://www.teamliquid.net/blogs/519872-towards-a-good-sc-bot-p56-latency

[3] http://www.starcraftai.com/wiki/Chance_to_Hit

[4] https://www.codeofhonor.com/blog/the-starcraft-path-finding-hack

[5] https://liquipedia.net/starcraft/Marine

*3.2.6 Army size.* The upper limit to the size of a player's army is bound by the physical limitation of the number of marines that can be placed in each others weapon range. The lower limit is partly inspired by Hu et al. [9]. In this work, a 5-versus-5 scenario is used, in which a 100% winrate is obtained by an AI implementation. Such a one-sided outcome due to the low number of units used is undesirable here, as the purpose of this experiment is to identify differences between the organisational paradigms. Variance in the outcome of the trials is thus desired; if all implementations reach a 100% winrate without actually requiring (complex) multi-agent teamwork, there is little to compare them on. Because it takes 7 shots to eliminate an opposing marine[6], it is more interesting to simulate a battle with at least more than 7 units in order to observe intelligent decision making on shot allocation and inter-agent coordination by the MAS. An army size of 10 fits within both the lower and upper limit, and is thus used as the size of both armies in this custom map.

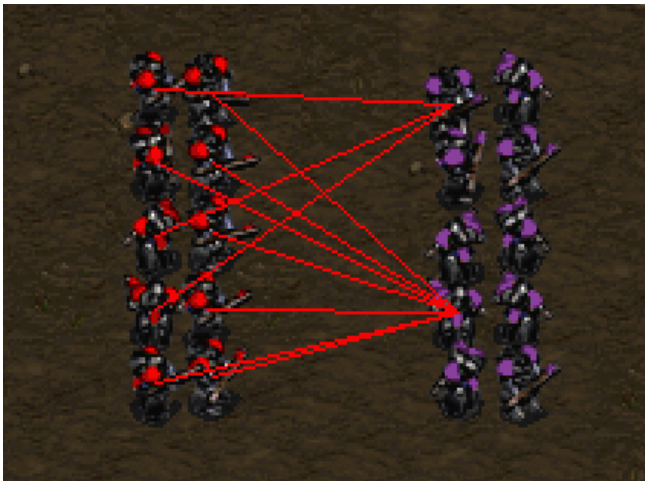

**Figure 1: Scenario setup. In red are the units controlled by the MAS, in purple the ones controlled by the built-in AI. The red lines illustrate the current target of each red unit.**

By considering all factors listed above, the variance that can arise from the scenario implementation is reduced. The outcome of a trial thus becomes more dependent on the type and implementation of the organisational paradigm used, and less trials will be needed in order to compare the impact of (only the) organisational paradigms.

## 3.3 Internal Randomness

One of the consequences of SC:BW not being a theoretical concept but a practical scenario is the fact that there are random factors introduced intentionally by the game in order to add variance (and thus a challenge for human players), which cannot be controlled for by using a custom map. There are three conditions that can influence the targeting scenario as described here, namely the initial orientation of units, their weapon cooldown after an attack, and the way in which the enemy AI is controlled.

[6]Each marine has 40 hit points, and marines deal 6 damage per shot.

*3.3.1 Initial Orientation.* When units initially spawn, they face a random direction. As they might face away from an enemy and need time to rotate towards that enemy, it can take up to 4 extra frames before they can shoot their first shot[7]. In order to quantify the impact of this random factor, in each trial the total initial number of frames spent rotating towards the enemy targets is measured for both the player and the enemy player. Section 6 analyses these values and determines their impact.

*3.3.2 Weapon cooldown.* When a *Marine* shoots, the weapon used will go on cooldown, i.e. the unit will not be able to attack for a certain number of frames. These cooldowns are slightly randomised by the game, and can differ $[-1, 2]$ frames when compared to each other. In order to prevent the weapon cooldowns from affecting the comparability of our results, the number of trials for each MAS was kept relatively high.

*3.3.3 Opposing army control.* The opposing army must always be controlled in the same way, as otherwise an additional independent variable would be introduced, i.e. the opponent's strategy. Through a setting in the map builder, control over the opposing army is given to the *built-in AI* of SC:BW. The enemy army is also subject to both the random initial orientation and weapon cooldown, but the control system that decides their strategy now remains constant.

## 3.4 Performance Strategy

The given combat scenario depends heavily on units killing enemy units, and thus reducing the opponent's total fighting power before the inverse happens. As all units are in combat range of each other, targeting is the only way to influence the outcome in this specific scenario. A good targeting strategy is the *No-OverKill-Attack-Value (NOK-AV [3])* targeting selection. This targeting mechanism aims to prevent wasting additional shots on a target once lethal damage has been (or will be) applied. This can be done by keeping track of damage inflicted for each unit and keeping track of their remaining health. The point of not overshooting (i.e., to not *overkill*) is important, as marines can only shoot one bullet at a time. If for example three *Marines* decide to simultaneously shoot a unit that could have been neutralised by one shot, there are effectively shots wasted that could have been allocated to other targets. Naturally, this problem only worsens for larger groups of units. The unit with the lowest health is the easiest to neutralise, and consequently decreases the overall force of the army fastest. For this reason, as specified by NOK-AV, the lowest-health unit is the highest-value target and thus prioritised, but by keeping track of the shots (going to be) fired on this target NOK-AV also ensures switching focus to another target once lethal damage has been (or will be) dealt.

# 4 METRICS

This section describes the performance metrics used for summarising the results of a performed trial of a MAS (using a specific organisational paradigm) in the scenario as described in the previous section, as well as the software engineering metrics used for

[7]The orientation of a unit is described with a value in the range of $0 \leq \text{orientation} < 256$. Marines have a turn rate of $\frac{40}{256}$ per frame. Given that they can turn both clockwise or counterclockwise, the worst-case turn duration is 4 frames.

measuring the software engineering aspects of the implemented agent systems.

### 4.1 Performance Metrics

In order to quantitatively measure the outcome in the described scenario, providing insight into the effectiveness of the organisational paradigm used for an implementation, the following aspects are taken into account:

(1) The number of surviving units ($-10 \leq n \leq 10$).
(2) The number of health points per surviving unit (array of values $0 \leq v \leq 40$).

For the first measurement, negative values indicate that the game was lost, and thus how many units of the enemy force survived. The number of surviving units is interesting on its own, as the ability to let more (low health) units survive could for example be preferred over having fewer (full health) units, depending on the situation.

From these two metrics, two additional aggregated metrics are derived that can summarise the trial in two quantifiers. The first is if the game is won (*1* for victory or *0* for defeat), based on the sign of the first metric. The second is the total remaining health, which is computed by taking the summation of the total amount of health points remaining for all units of the player ($-400 \leq n \leq 400$). This derived metric can highlight how much health the organisational paradigm has at the end of a trial, allowing for easy comparison between the different organisational paradigms.

### 4.2 Software Metrics

In order to compare the implementations of the different organisational paradigms, there is a need for measurements that can aggregate the written software into something comparable. A way to compare software is the use of software metrics, as they can be used in practise to help characterise software systems [15]. Three such metrics based on code statistic are: *Average Lines of Code*, *Average Number of Methods*, and *Average Cyclomatic Complexity*. Lanza and Marinescu [15] argue for the use of these metrics as they are independent (i.e., when one of these metrics change the others remain the same) and project independent (i.e., when the size of the project increases or decreases the metrics do not). However, in this work, we want to compare different implementation sizes too. Therefore, switching the statistic based method *Average Lines of Code* to *Total Lines of Code* provides more insight, as it relates the total size of the implementations. With that adaptation, the metrics are as follows:

(1) Total Lines of Code (LOC)
(2) Average Number of Methods (ANM)
(3) Average Cyclomatic Complexity (ACC)

In section 6, these metrics are reported and compared for all implementations.

## 5 MULTI-AGENT ORGANISATIONAL PARADIGMS

A vast number of theoretical models of organisational paradigms can potentially be applied in the described scenario [6, 11, 19]. In this paper, we focus on paradigms that differ in the level of autonomy and communication entailed by the organisational structure. The two factors are examined as illustrated in Figure 2. At the first 'lowest' level, all agents are autonomous **individual** entities that do not communicate or take orders. The next autonomous level allows communication between the entities, thus enabling agents to choose to align their actions with other agents. As there is no overarching entity in control, this is **swarm**-like behaviour. At the halfway point for both communication and autonomy is the **market**-based organisational paradigm. With jobs available for agents to do, agents have some freedom to decide what tasks they will perform, but ultimately they are taking orders as all tasks have been created by a controlling agent. The last model is the hierarchical paradigm with no autonomy and communications for its agents. In this 'fully **hierarchical**' (or top-down) paradigm, a tree-based structure is used in which each agent does as it is instructed by its parent node. We note that 'level of autonomy' as used here is similar to the inverse of the 'level of hierarchy' as used by Livingstone [17].

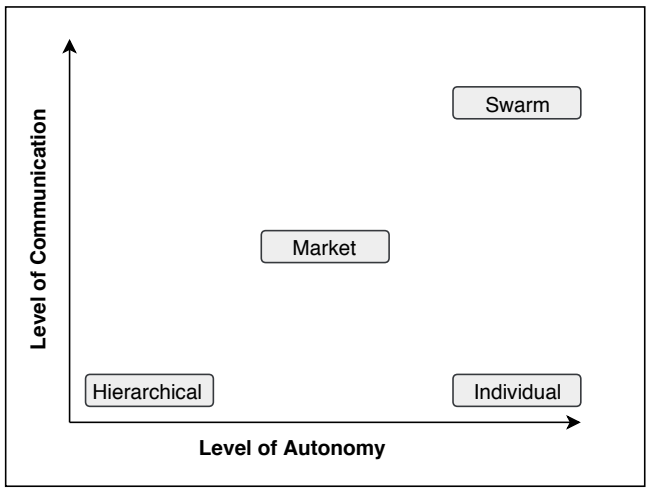


**Figure 2: An overview of the four organisational paradigms, illustrating the level of autonomy and level of communication inherent to each approach.**

### 5.1 Individual

The individual organisational model allows no communication between agents. Figure 3 displays the organisational structure of the Individual-based Multi-Agent paradigm. The main benefit of this style is that there is no overhead in maintaining this paradigms as agents are added or removed. Additionally, behaviour only needs to be implemented from the point of view of a single agent, and in a homogeneous setting, this behaviour can then be copied to all other agents. The downside is that as communication is not possible, the agents cannot easily coordinate to use team strategies. However, as agents can still perceive the environment, they can observe what their allied agents are doing and alter their own behaviour based on that in order to benefit the overall strategy.

As described in section 3.4, ideally agents target high-value targets first. This can be implemented by searching through the health values of all enemy units to find the unit(s) with the lowest amount

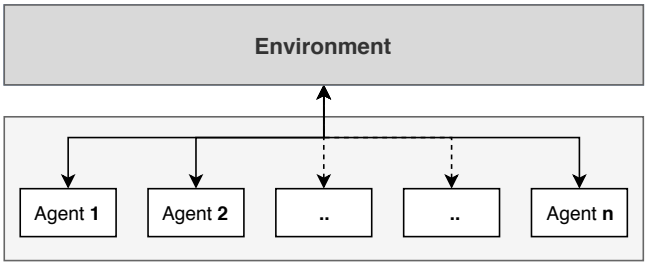


**Figure 3: Individual based Multi-Agent Paradigm**

of health points (HP). When all allied units employ this behaviour, they will thus focus on eliminating that unit. When all enemy units have the same HP, the agents can decide who to target either randomly or by the target locations to again show uniform behaviour.

The second part required to satisfy the performance strategy from section 3.4 is to not waste additional resources on a target that has already been defeated. Theoretically, the individual paradigm would be able to fully fulfil this constraint, as each unit can see if a target is still standing or not. In practice, however, the agents have to deal with a latency of two frames. This latency can cause shots to be wasted by the individual model if, at the moment an agent decides on a target, another one has already started preparing his attack with the lethal shot. An example of a situation in which this occurs is if there is an enemy marine with almost no health points remaining, and two allied marines are available to shoot. The first marine will fire the shot, and if communication was possible, the next one would receive an update of the game state and decide not to shoot the same target. However, as there is no communication in this paradigm, the second marine cannot know someone else is in the process of shooting the enemy marine. This causes the second marine to shoot the same target (at the same time) and consequently waste one bullet.

## 5.2 Swarm

The swarm-based organisational model assumes no agent is 'above the others' in the pecking order. All agents have equal control and autonomously decide what to do. The main benefit of this paradigm is that its decentralised approach is very resilient to the loss of agents in a harsh environment [25]. Due to agent casualties in the targeting scenario, SC:BW can be considered a harsh environment.

The downside is that because any agent can communicate with any other agent, from an implementation perspective this is a lot of extra work compared to the individual paradigm. In order to limit the complexity of the multiple communication channels, a swarm can be implemented using a blackboard [24]. With this method, each agent in the swarm synchronously gets access to the blackboard in a round-robin fashion [1]. First to read the state of the organisation, and then to decide and post what it will do based on the state of the swarm. Figure 4 displays the organisational structure where agents can both read and write to a collective state and interact with the environment.

In our scenario, each target unit is added to the blackboard if not seen before. For each target, the number of shots required to neutralise it is maintained and updated by the agents belonging

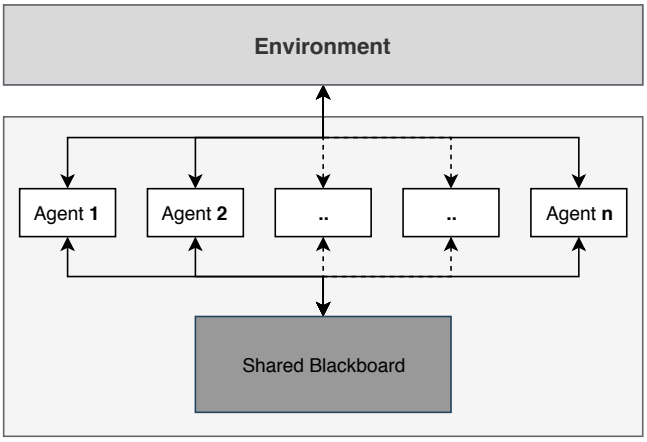


**Figure 4: Swarm-based Multi-Agent Paradigm**

to the swarm. Agents pick the target with the lowest health and then immediately decrease the health by the number of damage it can apply. Even with this simplified model, issues can still arise and influence the performance. For example, if an agent reads the blackboard and decides to shoot at a unit, but dies before being able to actually fire the shot, the rest of the swarm will assume this shot was fired.

## 5.3 Market

A newer and increasingly popular approach to hierarchical multi-agent systems is the so-called market-based approach. This approach has already found success in multiple fields such as controlling electricity grids [12, 14] and other load-based problems [10].

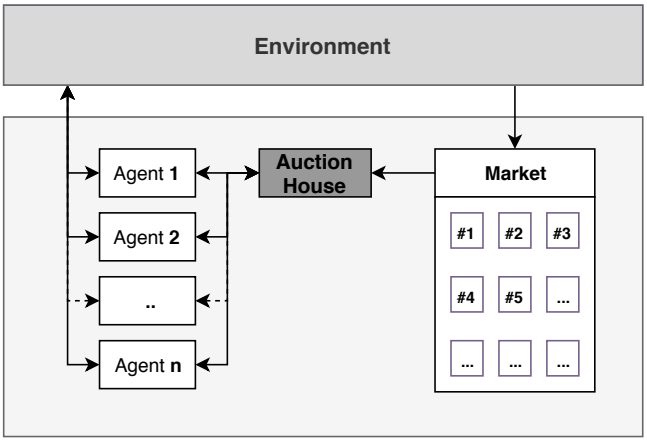


**Figure 5: Market-based Multi-Agent Paradigm**

In a market-based system, individual agents take decisions that benefit themselves the most, i.e. maximising their utility, but the choices they make are limited by what a central auction house has to offer [11]. This abstraction offers a method to define constraints on two levels: locally as an agent can choose what is the best for

it at that moment, and globally as the auction house can define its offers to benefit the task at hand. Furthermore, no communication between individual agents is required. Figure 5 displays this architecture in which the market observes the environment and creates available items or tasks that can be obtained. The auction house can then make these items available for bidding to the agents that control the marines, taking the overall strategy into account.

The market-based paradigm is modelled in our scenario as a single auction house selling 'shots' on enemy targets as the product (e.g., resource allocation), where low-health targets are offered before high-health ones. Individual agents can then bid on these products with an offer that scales with their weapon cooldowns. The auction house will sell the products to the highest bidders, and then start a new bidding round for the agents that did not get to buy a product if they wanted one. This continues until there are either no more products or no more buyers.

## 5.4 Hierarchical

The structure as seen in Figure 6, where a central Control Agent decides the strategy and commands the agents it has control over, is what we call the (fully) hierarchical model. In theory, this paradigm functions like a tree, and can have multiple levels. Various existing SC:BW AI bots make use of multi-layered hierarchical architectures that have a middle layer that groups agents into squads [21]. We did consider 'squads', which are similar to holarchies [5]. However, on the limited army size used, the paradigm would add more overhead than benefits.

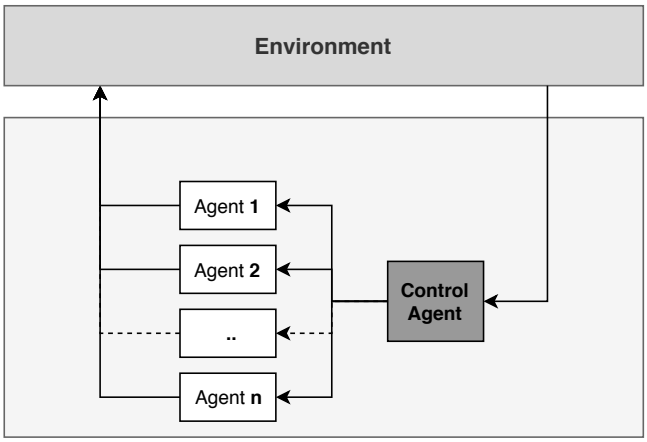


**Figure 6: Hierarchical based Multi-Agent Paradigm**

This experiment uses one level, where the control agent directly commands the agents connected to the marines in the environment. This top-down approach, together with the individual approach, is the most simple in terms of implementation at the start, as only one agent needs to be implemented. Only this agent needs to make decisions, and as long as the task and environment are confined to a single scenario, this offers a simple and effective abstraction. However, this structure does not quite scale, as a lot of decision making has to be done in one place. If, for example, the environment would have multiple tasks in multiple locations at potentially different time frames, a (top-down) hierarchical system quickly grows in terms of complexity.

The experiment in this paper, however, entails a single-task scenario, so it is valuable to compare the performance of this single-layer hierarchical organisational structure. For the implementation, the root node, i.e. the control agent, reads the whole environment. Based on the state of all its sub-nodes and the target units, the control agent selects the best targets for each individual agent (NOK-AV) and orders them to target them for the next shot. These sub-nodes are also agents, but they do not do any information processing of their own; their only information is what target the control agent ordered them to target. This approach does not require any communication between individual agents either, except listening to the control agent that is specifically introduced (similar to an auction house in the market-based approach).

# 6 RESULTS

This section described how the agent systems were evaluated and provides the results obtained from the experiment according to the metrics as described in section 4.

## 6.1 Preparation

In this work, we created entities as cognitive bodies around the in-game StarCraft units. These entities function as our agents, and each are responsible for the (rational) control of one in-game unit. Each agent is capable of executing tasks autonomously whilst communicating with the other agents. Agents that are not bound to in-game units are also used in order to reason about the state of the environment and (proactively) delegate MAS behaviour (e.g., the auctioneer in the market-based paradigm).

All multi-agent organisational paradigms have been implemented using Java 8 and are built on top of BWMirror [8]. There are a multitude of specialised agent programming languages such as Jason, 2APL, GOAL, and others [2]. However, obtaining software metrics from these specialised languages is not (yet) easily achievable. Java, however, has a long history in the software development world and has a multitude of source code analysis tools available. Moreover, interfacing (cognitive) agent systems to StarCraft is not a trivial task; the first work to this effect was published only very recently [13]. In order to obtain fair comparisons of the software metrics, each model is optimised with respect to those metrics.

Each system has full control over the allied units in the example scenario map as defined in section 3.2. This is achieved by playing on the game mode *Use Map Settings*, which is the game mode that supports custom maps. Each MAS implementation (and thus organisational paradigm) was run **1000** times. The results of each run were stored according to the metrics as defined in section 4. Every run has been executed on the same machine under identical conditions.

## 6.2 Performance Metrics

In Table 1, the aggregated results of the performance metrics are listed for each organisational paradigm.

The first column, $P_{win}$, shows the percentage of games won out of the 1000 games in total. Because a change in $P_{win}$ directly

[8] https://github.com/vjurenka/BWMirror

**Table 1: Performance metrics**

| Paradigm | $P_{win}$ | $\mu_{units}$ | $\sigma_{units}$ | $\mu_{hp}$ | $\sigma_{hp}$ | r-value |
|---|---|---|---|---|---|---|
| Individual | 0.927 | 2.71 | 0.99 | 73.72 | 30.18 | 0.132 |
| Swarm | 0.920 | 2.70 | 1.00 | 71.63 | 30.54 | 0.125 |
| Market | 0.951 | 3.22 | 0.99 | 94.21 | 31.60 | 0.119 |
| Hierarchical | 0.937 | 3.00 | 1.00 | 85.38 | 31.47 | 0.128 |

influences the numbers of the other columns, only the winning trials were used to compute the values displayed in those columns, thus allowing for objective comparisons between the paradigms. The mean ($\mu_{units}$) and standard deviation ($\sigma_{units}$) show the average number of units that survived a winning run. The next two columns, $\mu_{hp}$ and $\sigma_{hp}$, show the mean and standard deviation for the aggregated total health of all surviving units. Higher values indicate the organisational paradigm on average had more health points remaining at the end of each winning run. The final column displays the random orientation factor as discussed in section 3.3, which can be analysed for correlation between the $\mu_{units}$ to determine if the random initial orientation of each unit influenced the outcome of a trial.

*6.2.1 Performance.* The winrates of each implementation are within 4% of each other, but are significantly different, as determined by one-way ANOVA ($F = 5.847$, $p = .001$). In order of winrate, the swarm-based implementation had the lowest chance of winning with 0.920 and also has the lowest health remaining for all units at 71.63. The individual implementation has a similar winrate of 0.927 and a near identical average number of units surviving as the swarm-based implementation. The hierarchical implementation performs second best at a 0.937 winrate and also has a higher average amount of surviving units at 3 compared to the 2.7 of the swarm-based and individual implementations. The amount of surviving units is also significantly different for each organisational paradigm ($F = 58.842, p = 0$). The highest winrate is achieved by the market-based implementation at 0.951, together with the highest average number of units surviving. The third and final metric of average health also has significant differences ($F = 108.032$, $p = 0$). The market-based implementation has the highest average health remaining at the end of 1000 trials.

*6.2.2 Random Influence.* In order to measure how significantly the random factors as described in 3.3 influenced the results, the initial orientations were captured for each unit. The total number of frames required for turning towards the opposing army units was tallied for both armies, and the difference is taken as the *orientation difference* (e.g., if the MAS had to turn a total of 30 frames for all units, and the enemy had to turn 34 frames, the *orientation difference* is −4). The *orientation difference* and *the amount of health remaining* for the organisational paradigm of each trial are plotted against each other in a scatter-plot as seen in Figure 7. For each paradigm, the horizontal axis is the *orientation difference* and the vertical axis *the amount of health remaining*. Based on the r-value (the statistical value for the relationship between the two plotted variables), it is safe to conclude that there is no significant correlation between the random orientations and the outcome of a

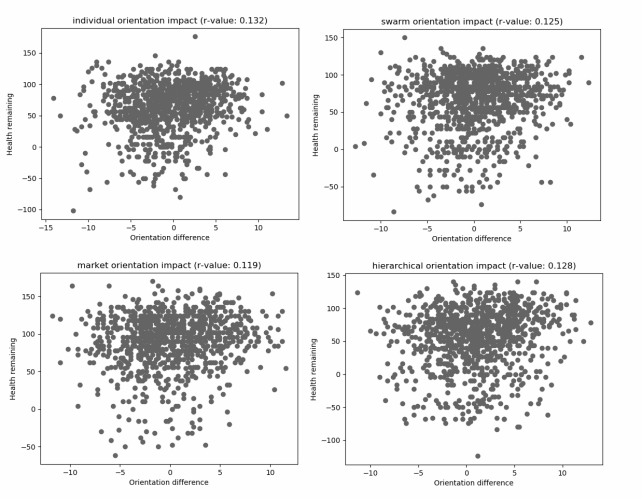


**Figure 7: Scatter plots of the orientation difference between both armies vs. the remaining health for the 4 MAS.**

trial[9]. As each organisational paradigm shows a similar r-value, the amount of influence this randomness introduces is, as expected, equal for each paradigm. Because the average measured orientation difference over all trials for each MAS was less than 0.2, this factor can be safely ignored.

## 6.3 Software Metrics

In Table 2, the collected software metrics are given for each organisational paradigm. The metrics were determined by using the program SourceMonitor[10].

**Table 2: Software metrics**

| Paradigm | LOC | ANM | ACC |
|---|---|---|---|
| Individual | 91 | 2.5 | 1.90 |
| Swarm | 137 | 3.0 | 1.56 |
| Market | 215 | 3.2 | 1.51 |
| Hierarchical | 152 | 2.3 | 3.78 |

For the total amount of lines of code (LOC), the largest difference is between the individual and market-based implementations with 91 and 215 respectively. The larger code size for the market-based implementation is inherent to its core design; the market-based implementation requires additional code for the auction house and the products it sells. In contrast, the individual implementation requires almost no extra code as only one agent needs to be implemented. The swarm-based and hierarchical implementations sit in between with 137 and 152 lines of code, where the blackboard and the controlling agent lead to the extra code that these implementations require respectively.

[9] A larger negative orientation difference value (e.g. −10) would indicate that the other army has to turn more, and thus waste time turning instead of shooting. If there was a strong correlation between turning time and the health remaining, we would expect to see the health decrease as the orientation difference increases. This can not be concluded from the scatter plots, so we can assume little to no correlation exists.
[10] http://www.campwoodsw.com/sourcemonitor.html

An independent-samples t-test shows that the average number of methods (ANM) metric results can be split into two groups ($t(2) = 4.95$, $p = .038$): the individual and hierarchical implementations with 2.3 and 2.5 methods on average, and the swarm-based and market-based implementation with 3.0 and 3.2 methods on average. This small difference likely comes from the fact that both the swarm-based and market-based implementations require data classes: the blackboard and the product respectively.

The ACC metric has one outlier: the hierarchical implementation, with a value of 3.78. This higher average complexity (ACC) is due to the control agent needing to track all of the agent states and making decisions for them. The market-based and swarm-based implementations, however, offer layers of abstraction that reduce the overall ACC, which explains their slightly lower ACC compared to the individual implementation.

## 7 DISCUSSION

In this paper, we implemented 4 different multi-agent systems based on well-known organisational paradigms from literature. These multi-agent systems where evaluated based on various performance and software metrics in order to give insight into the differences between the according organisational paradigms. Based on the results provided by the metrics, this section interprets and describes the identified differences between the organisational paradigms.

The market-based and hierarchical approaches use some kind of central processing agent that can calculate performance strategies using the larger picture, as it has knowledge and control of the field agents. Based on an independent-samples t-test on $\mu_{\text{units}}$ ($t(3691) = 12.299$, $p = 0$), it can be concluded that **when having a central control agent the MAS loses fewer units**, and so the chance of survival for an agent with less autonomy increases. However, when looking at software engineering metrics, it can also be concluded that these systems are more complex (in one way or another). The market implementation uses the most lines of code (215) and the hierarchical implementation has double the average cyclomatic compared to the other implementations (3.78). The hierarchical organisational paradigm becomes exponentially hard to scale when more factors, battles or units types are added as they will only increase the cyclomatic complexity further.

The main difference between the individual and swarm-based implementations is minor but crucial. Agents in the individual implementation base their actions on what they observe, whilst agents in the swarm-based implementation can rely on information being exchanged between them. The main challenge for individual approaches is dealing with the delay from observing the environment, whereas swarm-based approaches have the added complexity of handling communication. In this scenario specifically, with a minor latency of two frames, it can be concluded based on the very similar win percentages, units alive and unit health points remaining metrics of individual and swarm[11], that **there is no benefit to sharing target information and investing time into a shared blackboard**. With more latency, allied units have more frames in which they could falsely conclude that an already neutralised target is still alive. We therefore hypothesise that if the latency increases, the swarm-based organisational paradigm will perform better compared to the individual paradigm.

Another distinction can be made between the individual and hierarchical implementations compared to the swarm-based and market-based implementations. The former not having to deal with any communication (the hierarchical implementation does not entail interaction between agents except for unconditionally following orders originating from one source), whereas both the swarm-based and market-based approaches are based on collaboration that adds implementation complexity for the communication protocols that may or may not be worth the time (i.e., careful attention needs to be given to make sure tasks are distributed synchronously in order to prevent duplicate or unassigned tasks). When looking at the data, we can see that the variance $\sigma_{\text{hp}}$ is lower for the communication-less paradigms, i.e. when the complexity of communication is avoided. Moreover, independent-samples t-tests show that **the average amount of units** ($t(3691) = 3.363, p = .001$) **and the average amount of health** ($t(3691) = 3.436, p = .001$) **is significantly higher for the paradigms employing (inter-agent) communication**. However, such communication does not have a direct effect on the winrate; the hierarchical paradigm 'avoids' communication effectively by directly commanding all units. Finally, as seen from the software metrics, the market-based and the swarm-based implementations have a higher-than-average number of methods and contain more lines of code than the individual implementation.

In conclusion, the data suggests that for this scenario, **more autonomy or more communication does not necessarily help**. The middle ground approach of the market-based paradigm, however, stands out by having a better performance for all measured metrics, although at the cost of more implementation effort.

*Future Work.* The experiments as discussed in this paper are limited to one scenario. In future work, additional factors such as multi-tasking and adaptive capabilities could be explored. To this end, one could devise a scenario with multiple simultaneous battles. We predict that the individual and swarm-based solutions will require almost no modifications, the market-based implementation will need minor changes to the utility functions on the local and global level, and the hierarchical MAS will require a major rewrite or the addition of a new layer of controlling agents. At this scope, the transition to a holon-based approach could be beneficial.

Another approach would be to repeat the experiment in similar scenario in another RTS game, or another similar environment with either more or less latency. This could (dis)prove our hypothesis about the effect of this latency on the swarm-based approach.

Finally, when tools that can calculate software metrics on (cognitive) agent programming languages are available, re-implementing the four multi-agent systems in one of these languages could provide more insight into the effects of using such dedicated languages.

[11] An independent samples T-test indeed shows no significant differences between individual and swarm on any individual metric.

## REFERENCES

[1] Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau. 2014. *Operating systems: Three easy pieces.* Vol. 151. Arpaci-Dusseau Books Wisconsin.
[2] Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni. 2009. *Multi-Agent Programming.* Springer.
[3] David Churchill, Abdallah Saffidine, and Michael Buro. 2012. Fast Heuristic Search for RTS Game Combat Scenarios. In *Proceedings of the Eighth AAAI Conference*

*on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'12)*. AAAI Press, 112–117.

[4] Daniel D. Corkill, Daniel Garant, and Victor R. Lesser. 2016. Exploring the Effectiveness of Agent Organizations. In *Coordination, Organizations, Institutions, and Norms in Agent Systems XI*, Virginia Dignum, Pablo Noriega, Murat Sensoy, and Jaime Simão Sichman (Eds.). Springer International Publishing, Cham, 78–97.

[5] Massimo Cossentino, Carmelo Lodato, Salvatore Lopes, Patrizia Ribino, and Valeria Palermo. 2015. Metrics for Evaluating Modularity and Extensibility in HMAS Systems. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 1061–1069.

[6] J. Ferber and O. Gutknecht. 1998. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings International Conference on Multi Agent Systems (Cat. No.98EX160)*. IEEE, 128–135.

[7] Koen V. Hindriks. 2014. The Shaping of the Agent-Oriented Mindset. In *Engineering Multi-Agent Systems*, Fabiano Dalpiaz, Jürgen Dix, and M. Birna van Riemsdijk (Eds.). Springer International Publishing, Cham, 1–14.

[8] Bryan Horling and Victor Lesser. 2004. A Survey of Multi-agent Organizational Paradigms. *Knowledge Engineering Review* 19, 4 (Dec. 2004), 281–316.

[9] Yue Hu, Juntao Li, Xi Li, Gang Pan, and Mingliang Xu. 2018. Knowledge-guided Agent-tactic-aware Learning for StarCraft Micromanagement. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*. AAAI Press, 1471–1477.

[10] Kai Huang, Sanjeev K. Srivastava, David A. Cartes, and Li-Hsiang Sun. 2009. Market-based multiagent system for reconfiguration of shipboard power systems. *Electric Power Systems Research* 79, 4 (2009), 550–556.

[11] Nicholas R. Jennings, Katia Sycara, and Michael Wooldridge. 1998. A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems* 1, 1 (Jan. 1998), 7–38.

[12] Hak-Man Kim, Tetsuo Kinoshita, and Myong-Chul Shin. 2010. A Multiagent System for Autonomous Operation of Islanded Microgrids Based on a Power Market Environment. *Energies* 3, 12 (2010), 1–19.

[13] Vincent J. Koeman, Harm J. Griffioen, Danny C. Plenge, and Koen V. Hindriks. 2018. StarCraft as a Testbed for Engineering Complex Distributed Systems Using Cognitive Agent Technology. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, USA, 1983–1985.

[14] J. K. Kok, C. J. Warmer, and I. G. Kamphuis. 2005. PowerMatcher: Multiagent Control in the Electricity Infrastructure. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '05)*. ACM, New York, NY, USA, 75–82.

[15] Michele Lanza and Radu Marinescu. 2010. *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems* (1st ed.). Springer Publishing Company, Incorporated.

[16] R. Lara-Cabrera, C. Cotta, and A.J. Fernández-Leiva. 2013. A review of computational intelligence in RTS games. In *2013 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*. 114–121.

[17] Daniel Livingstone. 2005. Coevolution in Hierarchical AI for Strategy Games. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence and Games (CIG '05)*. IEEE.

[18] Linus J. Luotsinen, Joakim N. Ekblad, T. Ryan Fitz-Gibbon, Charles Andrew Houchin, Justin Logan Key, Majid Ali Khan, Jin Lyu, Johann Nguyen, Rex R. Oleson, Gary Stein, Scott A. Vander Weide, Viet Trinh, and Ladislau Bölöni. 2007. Comparing Apples with Oranges: Evaluating Twelve Paradigms of Agency. In *Programming Multi-Agent Systems*, Rafael H. Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni (Eds.). Springer Berlin Heidelberg, 93–112.

[19] Nelson Minar, Rogert Burkhart, Chris Langton, and Manor Askenazi. 1996. *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*. Working Papers. Santa Fe Institute.

[20] Jörg P. Müller and Klaus Fischer. 2014. Application Impact of Multi-agent Systems and Technologies: A Survey. In *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*, Onn Shehory and Arnon Sturm (Eds.). Springer Berlin Heidelberg, 27–53.

[21] S. Ontañón, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. 2013. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in Games* 5, 4 (Dec. 2013), 293–311.

[22] Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. 2015. RTS AI Problems and Techniques. In *Encyclopedia of Computer Graphics and Games*, Newton Lee (Ed.). Springer International Publishing, Cham.

[23] Glen Robertson and Ian Watson. 2014. A review of real-time strategy game AI. *AI Magazine* 35, 4 (2014), 75–104.

[24] J. Straub. 2015. Swarm intelligence, a Blackboard architecture and local decision making for spacecraft command. In *2015 IEEE Aerospace Conference*. 1–7.

[25] E. Vassev, R. Sterritt, C. Rouff, and M. Hinchey. 2012. Swarm Technology at NASA: Building Resilient Systems. *IT Professional* 14, 2 (March 2012), 36–42.

[26] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. 2017. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint arXiv:1708.04782* (aug 2017).

[27] Ben George Weber, Michael Mateas, and Arnav Jhala. 2011. Building Human-Level AI for Real-Time Strategy Games. In *AAAI Fall Symposium: Advances in Cognitive Systems*, Vol. 11. AAAI, Palo Alto, CA, USA.