# Multiagent-based decision making in strategy games

Diana Fernández-Tirado[1], Adolfo Pacheco-Nieto[2] and Gilberto G. Flores-Vidaña[3]

Instituto Tecnólogico de Estudios Superiores de Monterrey, Monterrey, N.L., México.
[1]A00746263@itesm.mx, [2]A00812892@itesm.mx, [3]A00735176@itesm.mx
*Paper received on 25/07/12 , Accepted on 06/09/12.*

**Abstract.** The video game industry strategy aims to generate more intelligent games, where characters have the ability to make decisions and reach agreements in order to increase the challenge and keep gamers captives. This article proposes a negotiation and decision-making model for agents in a strategy game based on Contract Net, and shows how an approach based on Multi-agent System and Contract-Net can have better results than an independent agent. The system analysis and design was modeled using GAIA methodology in which we can describe the interactions and behavior of the agents.

**Keywords:** Multi-agents Systems, Contract-Net, GAIA, strategy game.

## 1 Introduction

In the area of manufacturing we have process planning and scheduling that are two, usually considered, different activities that are handled by techniques of Distributed Artificial Intelligence [6] such as Multi-Agent Systems. There is extensive research around these problems as we can see in the survey presented by Shen [7] whereas in Lu [8] proposes a novel agent bidding based approach that searches for optimal solutions in manufacturing planning and scheduling. Decision-making and implementation of agreements in games of strategy is a challenge that can be addressed with Multi-agent techniques like Contract-Net.

The problem addressed in this article is to increase the intelligence of the Knights and Workers (as agents in the game) in the following scenario: the scene shown in the screen is a map, a civilization is located in an arbitrary point and starts with few Workers and the MainBuilding. In order to create more Buildings and Workers or train Knights, the civilization needs resources (gold and wood), Workers can obtain these resources from mines and trees. The Workers also can create buildings that are able to defend the civilization from enemies' attacks. All around the map there are enemies, that can be walking around or staying in a certain point, these enemies can assemble and decide (or not) to attack the civilization that is being built in the map. If the enemies destroy the MainBuilding of the civilization, the game is over. The Knights can improve their strength, when a Knight attacks an enemy it gains experience and eventually it is able to kill stronger enemies. The goal


*Advances in Intelligent and Information Technologies.*
*Research in Computing Science 60, 2012, pp. 297 -308.*

is to build a strong civilization and kill the MightyDragon that resides located in the map.

This paper shows the process to implement a Multi-agent System using the GAIA methodology to describe interactions between agents and Contract Net protocol for negotiation between agents, the project presents a scheme based on utilities in the strategy of Knight-Teamwork (in pursuit of the goal of killing the Mighty Dragon).

The remaining of this paper is organized as follows: section 2 presents the use of GAIA methodology in the system analysis and design of the game, in section 3 the prototype (developed in Net Logo) is described and a formula to calculate the utility used for decision-making in the Contract Net protocol is proposed, in section 4 the experiments made for this project along with the results obtained are detailed, and finally in section 5 we draw our conclusions.

## 2 Analysis and Design

Wooldrigde, Jennings and Kinny [4] argue that the traditional models for software analysis and design are not suitable for agent-based systems because they are unable to represent unique aspects of agents. They present the GAIA methodology, which has been tailored specifically for these kinds of systems.

The GAIA methodology is a process that leads the developer to go from the requirements to a design with such a detail that can be implemented straight [4]. Gaia has two main concept categories: abstract and concrete, the result of this process is: (1) during the analysis stage the roles and interaction models are defined and (2) during the design stage the agent, services and acquaintance models are described.

### 2.1 System Organization

In the game named “Knights and Dragons” the environment is composed by the map (in which the game will be developed) with the resources located all around the map (mines and trees), and the goal is to build a civilization strong enough to kill all enemies in the map.

Raising and protecting a civilization is a complex process that needs to be handled by several kinds of agents. In order to achieve the goal of the civilization, agents must agree how to interact, and thus will allow them to get the highest benefit to the civilization.

The agents are related to their environment because the map is where the civilization will develop, and the agents (Workers and Knights) will be scouting around the map in order to find more resources that will allow the civilization to keep growing, get stronger, defend against enemies and kill them. The general structure of the agents is shown in Figure 1.

The game starts with the MainBuilding, which will create Workers in order to start gathering resources needed to create buildings and increase the army. To create more Workers we need to check if there is enough food, otherwise more Huts are needed, in this way the civilization does not get limited by food.

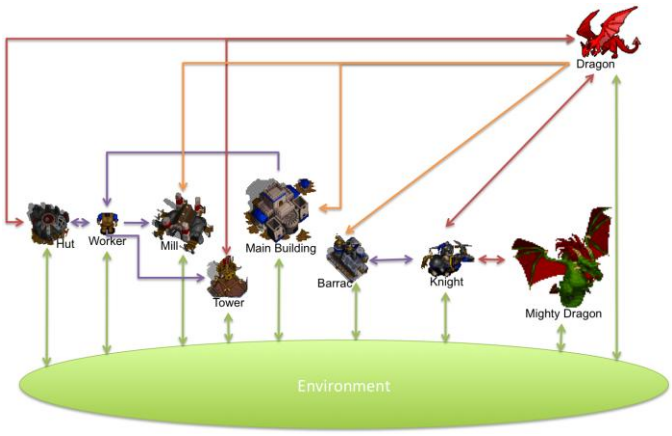


**Figure 1:** General Structure of Agents.

The game starts with the MainBuilding, which will create Workers in order to start gathering resources needed to create buildings and increase the army. To create more Workers we need to check if there is enough food, otherwise more Huts are needed, in this way the civilization does not get limited by food.

The Workers will start gathering resources or creating buildings, and if they require to create a building they need to ask for resources to the MainBuilding and this one will let them know if there are enough resources for the new building.

As soon as the Barrack gets built, it can start creating Knights, it has to be aware of how many Knights had been created, so the civilization does not have Knights they do not really need (yet).

The Knights will start scouting all around the map and if they find an enemy they will call for help from other Knights.

Buildings like the Hut and the Tower will help defend the town if enemies attack, when they notice enemies' attacks, the Workers will get into Huts in order to protect themselves and at the same time attack enemies.

The Dragons will be all around the map, they can be protecting a gold mine or just flying around the map, they can get to the town and attack, and even can agree with other Dragons to attack.

The Mighty Dragon is the strongest enemy in the map, the Knights have to find it and kill it, but as it is so strong they will need first to gain experience by killing other Dragons of different levels.

## 2.2 Roles Description

In the context of Gaia methodology a role is an abstract description of an entity's expected function and it is defined by four attributes: protocols, permissions, activities and responsibilities [3]. The role model states the roles an agent can perform in the environment described earlier. Role description, activities, protocols, responsibilities and permissions for the MainBuilding are presented as a significant example in Table 1.

**Table 1:** Schema for Role MainBuilding.

| ROLE SCHEMA: MainBuilding | |
|---|---|
| **Description:** Is the base of the town, it keeps the gold and the wood. This building can create Workers, and must decide if there are enough Workers or needs to create more. If this building is destroyed the civilization cannot survive. | |
| **Activities:**<br>• CheckGoldStock<br>• CheckWoodStock<br>• CheckFoodLimit<br>• Die<br>• CreateWorker | **Protocols:**<br>• IncreaseWoodStock<br>• IncreaseGoldStock<br>• RequestHutCreation<br>• GetResource<br>• GetHost<br>• ReceiveAttack |
| **Permissions:**<br>**Reads**<br>• IsCreatingWorker // TRUE \| FALSE<br>• IsInRequestForHutCreation // TRUE \| FALSE<br>**Changes**<br>• FoodLimit // Real Number<br>• GoldStock // Real Number<br>• WoodStock // Real Number | **Responsibilities:**<br>**Liveness:** (([Upgrade] \| [RequestHutCreation] ) \|\| (GetHost . GetResource . CreateWorker ) \|\| (CheckGoldStock . CheckWoodStock . CheckFoodLimit . IncreaseWoodStock . IncreaseGoldStock) \|\| ([ReceiveAttack] . [Die]) ) w<br>**Safety Properties:**<br>• This building must survive<br>• Create Workers when the town requires them<br>• Gold Stock > 50 and Hut Space > 1<br>• Gold Stock > 0<br>• Wood Stock > 0<br>• Workers > 0<br>• Health points > 0 |

## 2.3 Interaction Model

The interaction model is described in GAIA as the interactions between the roles that lead them to reach their goal [3]. Examples of the protocols used to describe interactions between agents are:

| **IncreaseGoldStock** | | |
|---|---|---|
| Worker | MainBuilding | LevelOfGold |
| After the Worker collects the gold, it can carry it from a mine; it goes to the MainBuilding or Mill and deposits the resource. | | GoldStock |
| **GetResource** | | |
| AbstractBuilding, Barracks, Hut | MainBuilding | AmountOfGold, AmountOfWood, Host |
| A building asks for resources to the MainBuilding to create elements (Workers, Towers). | | WoodStock, GoldStock, FoodLevel |
| **Attack** | | |
| Knight, Dragon, Tower MightyDragon, Hut, | Knight, Dragon, Tower MightyDragon, Hut, | Enemies |
| When an enemy is in attack range of the unit, the unit | | |

| | | |
|---|---|---|
| attacks the enemy. | | |
| **AskForHelp** | | FindEnemies, AttackFromEnemy |
| Knight | Knight | |
| When a Knight find one or more enemies or it is attacked by enemies, the Knight calls for other Knights to help him. | | |

### 2.4 Activities Description

Role's activities are actions that an agent can perform, and can imply interactions with other agents or not [3]. The activities realized by the roles described in the document are:

**CreateWorker**: When the MainBuilding detects that more Workers are needed, if the civilization has the amount of gold needed to create a Worker and there are enough Huts, the MainBuilding may decide to create a Worker.

**CheckForEnemies**: This activity can be done for some buildings, and it allows to know the existence of enemies nearby, so defensive or offensive actions can be taken.

**CheckFoodLimit**: The MainBuilding validates the maximum amount of food allowed by total Huts.

**CheckFoodLevel**: The MainBuilding takes control of the food level that exist in all the available Huts.

**CheckGold/CheckWoodStock**: The MainBuilding or Mill verifies constantly the amount of gold/wood that the civilization has.

**DefendMine**: This activity is used by the Dragons to protect a gold mine from Knights. And this activity force the Dragons to stay in the mine waiting if a group of humans appear.

**Die**: This activity can be used by any role; the role that uses it disappears of the game.

**Ignore**: This activity is used by the MightyDragon to ignore the Workers nearby.

**Scouting**: The Knights explore the map in order to find wood, mines or Dragons.

**Runaway**: This activity takes place if a Dragon or a Knight thinks it is better to flee than to attack.

**Walking**: This activity is used by the Dragons to explore the map if they aren't protecting a mine.

### 2.5 Agent Model

During this step of the design stage we will identify “the agent types that will make up the system, the agent instances that will carry out these agent types at run-time, and the mapping between roles and agent types”[3].

The agent model (Figure 2) shows which roles are associated with each agent and how many agents of each type may exist in software. This model indicates that there is only one TownHallAgent because the game objective is to keep it safe. In

the simulation strategy game the creation of a set of DefensiveTowerAgents is needed to defend the town from the Dragons' attacks. To increase the FoodLimit of civilization is important to create several HutAgents. There is only one agent who creates KnightAgents, the BarracksAgent. The MillAgents are used to gather wood and gold.

During the game there will be several WorkerAgents that create the buildings, collect gold and wood, besides defending the people using the HutAgent. The KnightAgents are responsible for performing the exploration of the map and release the mines from DragonAgents; there may be several Knights in the game. The DragonAgents protect mines or walk around the map. Finally there must be only one MightyDragonAgent because its strength is higher than the other Dragons.

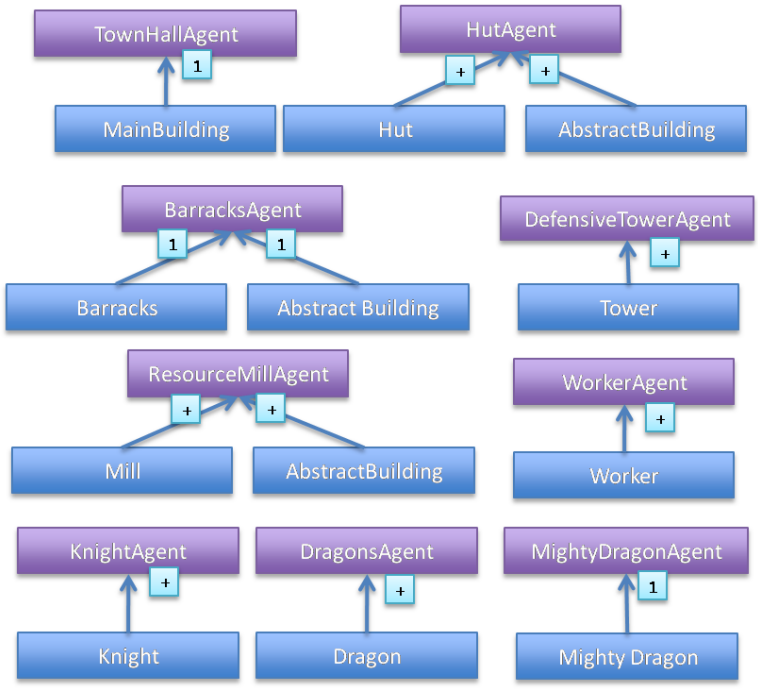


**Figure 2:** Agent Model.

## 2.6 Acquaintance Model

The goal of this step of the process is to document the communication lines that exist between the agents [3]. This model presents the existing interaction between agents (Figure 3), which are grouped in two main categories: buildings and characters. In the buildings category we can find MainBuildingAgent, the main function of this agent is to create Workers and keep gold/wood. In order to increment defenders for the town, the BarracksAgent creates and trains Knights. In the civilization the WorkerAgent can create TowerAgents, which can attack enemies and protect other buildings.

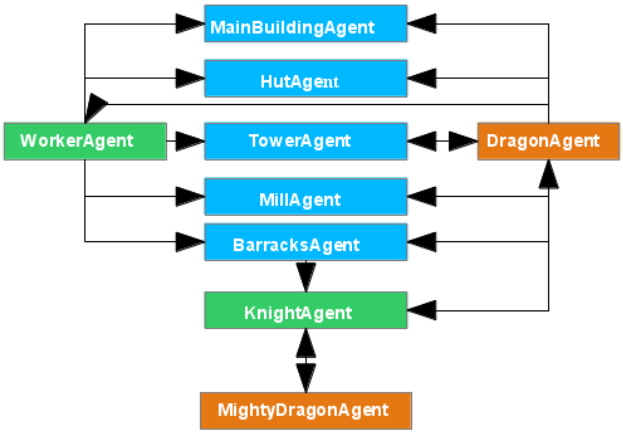


**Figure 3:** Acquaintance Model.

The characters category is compound of the WorkerAgents, KnightAgents, DragonAgents and MightyDragonAgent. The main interactions for WorkerAgent are the recollection of resources and creations of buildings. The KnightAgent attacks enemies, defends the town and goes scouting. MightyDragonAgent attacks the KnightAgents that are around it.

## 3 Prototype description

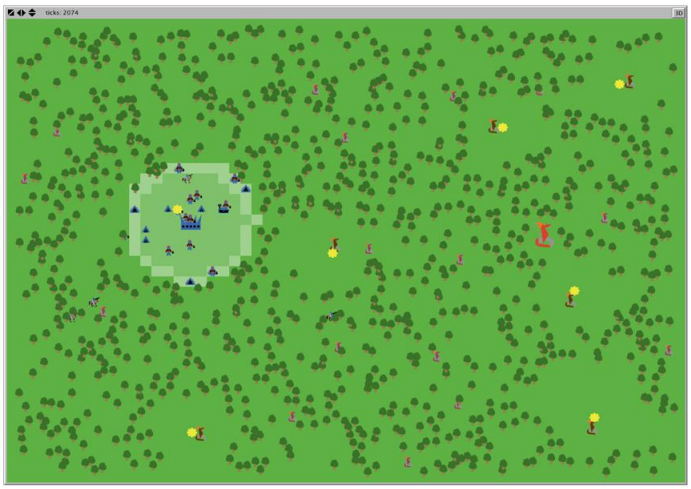

**Figure 4:** Screen of the software civilization simulator.

The Prototype for this article was implemented using Net-Logo. The prototype includes the following features:

**Civilization**: There are rules about the civilization, resources and building generation. Those are described below:

- The civilization must be established, and grow near a mine. In order to start mining a new mine, a Mill must be built near this new mine.
- Free mines will be guarded by enemies, in order to start getting gold from a new mine Knights need to kill them, so the Workers can start mining it.
- The Workers will be harvesting wood, and the trees will be cut off.
- The civilization will have a max food limit; this means how much Workers and Knights the civilization is able to feed. The Huts help improve the actual food limit (1 Hut = 10 food), but there is a top limit.

The game simulator is shown in Figure 4.

The user interaction interface has controls for:

- Max Food Limit.
- *Mines:* Maximum number of mines around the map and the maximum number of Workers that can be mining (a same mine).
- *Trees:* Maximum number of trees to be placed in the map.
- *Worker:* Gathering capacity, creation cost and initial number of Workers.
- *Knights:* Training cost, health points and strength.
- *Buildings:* Gold and lumber required to create different type of buildings.

### 3.1 Contract-Net implementations

Contract Net is a high-level protocol for communication among software agents (nodes) in a distributed problem solver that facilitates the task-sharing[2], that means It allows tasks to be distributed among a group of agents. For Contract Net there are two different types of agents, an Initiator and a Participant. At any time, any one agent can be an Initiator, a Participant or both [5]. Negotiation is implemented as two-way communication, in which an agent evaluates the offer of assigning a contract or receiving one from its own perspective [1], the protocol describes a number of basic messages that provide the necessary functionality for contract to be formed between agents.

Some of these messages are:

- **Task Announcement:** An agent does a call in which reports a task, with a complete specification of it, to be achieved.
- **Bid:** Agents, who listen had the call, decide for themselves if they are capable to perform or support the task and submit a tender.
- **Award:** Agent who did the call, choose between bids and selects who to award the contract.

The prototype implements four scenarios that apply Contract-Net:

**Building construction team.** In this scenario a Worker can request assistance from other(s) Worker, to be able to make buildings more quickly.

**Knight's Teamwork.** When a Knight finds a Dragon on the map, it asks for help to other Knights to attack the Dragon. Once the leader Knight estimates it is feasible to kill the Dragon, they go back and attack it.

**Dragon Teamwork.** When Dragons get to the Town, they ask other Dragons nearby to help them destroy the civilization.

**Workers defend in the Huts.** When Dragons attack the village, the Workers go inside the Hut and the defense of the village starts.

### 3.2 Knight decision-making

We calculate the utility of the Knights who fought against a Dragon using the following formula:

$$Utk = WHp \times Hp + Wstr \times Str - Wd \times d$$

In which:

- Utk: Utility of the group of Knights.
- WHp: Represents the Weight for Health Points
- HP: Health Points of the Knight.
- Wstr: Is the Weight to balance the strength of one Knight.
- Str: The current strength of a Knight
- Wd: The distance weight between the manager and potential contractor.
- d: The distance between the manager and potential contractor.

The use for this formula is shown in the Figure 5, with the following weights: Whp = 0.5, Wstr = 0.7, Wd = 50. The circles with letter K represent the KnightAgents, and the cycles with letter D represents the DragonAgent. The gray circle represents the vision of the Knight. The results for this scenario were: Utility for $K_1$ = 70, Utility for $K_2$ = -157 and Utility for $K_3$ = 34. The order in which the Knight will choose the others is: $K_1$, $K_3$ and $K_2$.

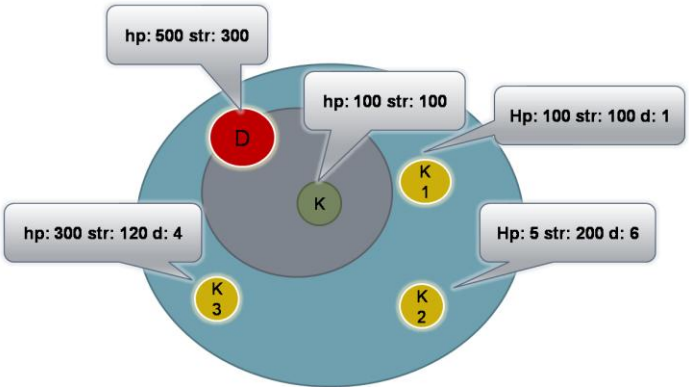


**Figure 5:** Selecting Knights.

However it is not sufficient to consider only the utility to attack the Dragon. To make better decisions is important to evaluate if the Knights who are in the contract are able to eliminate the Dragon. To resolve this problem, the following formulas were designed:

**Number of attacks by Knight**

$$Ak = \frac{Hp}{Hit_D}$$

**Sum of Ak of all Knights.**

$$S = \sum_{i=0}^{n} Ak_i$$

**Total damage utility for Knights**

$$\boldsymbol{DUK = \sum_{i=1}^{n} \{[W_o Ak + W_p (S - Ak)] Hit_i\}}$$

**Damage Utitily for Dragon.**

$$DUD = \frac{Hp \times Hit_D}{\sum_{i=1}^{n} Hit_i}$$

Where:

- n is the number of total Knights that accept the contract.
- Hp: Help points
- Hit: Represents a percentage of the agent strength. If this percentage is higher, the damage generated by the agent will be higher.
- $W_o$: This factor can appreciate how important it is to calculate the chances of attack that has a knight.
- $W_p$: Indicates the probability that a knight can attack more times than allowed by their Health Point. If this value is closer to 1 the system assumes that the agent will have a greater opportunity to attack.

If DUK > DUD then the Kngiths will attack, otherwise they run away.

## 4 Experiments and Results

The experiments compare the four types of Contract-Net strategy versus a random behavior of the agents. The first test does not include any Contract-Net, the behavior of the agents was random. As expected, the game consumes a lot of ticks and commonly the dragons win. The civilization can withstand using the TownHall and some of the knights that it can create. The results for this scenario are shown in the Table 2.

**Table 2:** Random Strategy

| Result | Ticks | Spent Lumber | Lumber Harvested | Spent Gold | Gold Mined |
|---|---|---|---|---|---|
| **Lose** | 40,371 | 2,880 | 3,052 | 12,950 | 11,980 |
| **Win** | 36,913 | 18,390 | 21,245 | 58,445 | 84,460 |
| **Lose** | 41,106 | 3,560 | 3,690 | 14,015 | 13,040 |
| **Lose** | 62,815 | 3,205 | 3,410 | 12,940 | 11,980 |
| **Lose** | 54,427 | 3,730 | 5,300 | 12,915 | 11,960 |

.

The second scenario uses a basic contract-net schema for dragons in which some dragons can create groups and attack the town. Using this strategy the dragons reduce the number of ticks in which they destroy the civilization. The results for this test are shown in the Table 3.

The last scenario adds three contract-net strategies for the civilization: Building construction team, Knight's Teamwork, and the defense of Workers in the Huts. This experiment shows the advantage for the civilization, the Knights can defeat the dragons every time and reduces the number of ticks. The knights can free some mines

and the workers can mine more gold. In one of the experiments the civilization kills the MightyDragon in only 3908, increasing its efficiency and winning all the games. The results for this scenario are shown in Table 4.

**Table 3:** Contract Net Dragon Team Work.

| Result | Ticks | **Spent Lumber** | **Lumber Harvested** | **Spent Gold** | **Gold Mined** |
|---|---|---|---|---|---|
| **Lose** | 16,399 | 2,710 | 2,784 | 12,960 | 11,980 |
| **Lose** | 25,384 | 3,675 | 4,182 | 13,285 | 12,300 |
| **Lose** | 23,588 | 2,205 | 2,210 | 12,940 | 11,950 |
| **Lose** | 27,472 | 3,380 | 3,680 | 14,470 | 13,480 |
| **Lose** | 29,310 | 3,630 | 4,992 | 12,975 | 12,020 |

**Table 4:** All Contract Net Strategies.

| Result | Ticks | **Spent Lumber** | **Lumber Harvested** | **Spent Gold** | **Gold Mined** |
|---|---|---|---|---|---|
| **Win** | 9638 | 3640 | 3661 | 11245 | 15790 |
| **Win** | 3908 | 2780 | 2831 | 7535 | 6920 |
| **Win** | 8849 | 3575 | 3662 | 10280 | 18750 |
| **Win** | 9958 | 3190 | 3381 | 9470 | 15820 |
| **Win** | 6779 | 3900 | 4280 | 11730 | 17030. |

## 5 Conclusions

The application of methodologies for the development of Multi-Agent Systems requires an analysis focused on roles and protocols that seek to describe the activities and forms of interaction of agents that function as autonomous units with ability to interact and reach agreements. In contrast to the development schemes of traditional systems, this approach requires a radical shift in the process of analysis and design, which seeks to increase efficiency in the coordination of tasks.

For strategy games the achievement was to bring more intelligence to agents that perform the activity for which they were created, because of the use of techniques like Multi-Agent Systems, Contract-Net made possible that agents were able to increase efficiency, improve effectiveness and reduce significantly the time when they perform a specified task.

This work shows the basis of coordinated interaction in Multi-Agent Systems; however, it is necessary to increase the decision-making capabilities of the agents at the time of reaching an agreement with the goal of generating more intelligent strategy games and with a close connection with the reality.

### References

[1] Chad La Fournie (2003) Cooperation using Negotiations and the Contract-Net Protocol, Department of Computer Science, University of Calgary Canada.

[2] Reid G. Smith (1980) Contract Net Protocol: High-Level Communication adn Control in a Distributed Problem Solver, IEEE Transaction on Computers Vol. c-29

[3] Villarreal, P., Alesso, M., Rocco, S., Galli, M.R, and Chiotti, O. (2003). Approaches for the Analysis and Design of Multi-Agent Systems. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.21, pp. 73-81.
[4] Wooldridge, M., Jennings, N., Kinny, D. "The Gaia Methodology for Agent- Oriented Analysis and Design". Journal of Autonomous Agents and Multi-Agent Systems. Vol. 3, no 3, 2000.
[5] Zafeer Alibhai (2003) What is Contract Net Interaction Protocol?, IRMS Laboratory, SFU
[6] J. Madejski (2006) Survey of the Agent-Based Approach to Intelligent Manufacturing, Silesian University of Technology, Gliwice Poland.
[7] Weiming Shen, Lihui Wang, Qi Hao, "Agent-Based Distributed Manufacturing Process Planning and Scheduling: A State-of-the-Art Survey". IEEE Transactions on Systems, Man and Cybernetics, Vol. 36, No 4, July 2006.
[8] Ruiqiang Lu, David Zhengwen Zhang (2010), "A Novel Agent Bidding Based Optimization Approach in Manufacturing Plannig and Scheduling", University of Exeter, Exeter United Kingdom.