Towards a Method for AI Problem Modeling in Real Time Strategy Games

Fernando Rocha Thiago Andrade Geber Ramalho Patrícia Tedesco

Federal University of Pernambuco, Informatics Center, Brazil

Abstract

Modeling and implementing game AI is often a complex task in game development, particularly when dealing with real-time AI which is the case of Real-Time Strategy Games (RTS). Most of current literature on game AI only proposes the application of AI to specific problems. In this paper, we introduce an original method devoted to help the developer to model the game AI. The method is the result of a five-year effort of observing and coaching under-graduate students to implement the AI of Non-Player Characters (NPC) for RTS games.

Keywords:
Real-Time Strategy Games, Artificial Intelligence, Game AI

Authors' contact:
{fafr,tas3,glr,pcart}@cin.ufpe.br

1. Introduction

It is known that computer games are considered a fairly complex computing environment, where, in addition to the whole process of control, one needs to consider also problems related to Artificial Intelligence (AI). Some genres such as Real-Time Strategy Games (RTS) are still more complex, since their environment involves several variables, each one of these with the potential to influence each other and the environment itself [Bourg 2004][Weber 2011]. The decisions made in RTS must take time into account to combine tactics and strategy adequately. Finally, some RTS games can also have large number of agents controlling Non-Player Characters (NPC) [Walther 2006].

The process of developing Game AI has often three stages [Rollings 2003]: modeling, implementation and adjusting. In the modeling stage, the developer must analyze the game in order to identify and map all the problems which require AI, as well as the relevant variables involved in these problems, the reasoning approaches that can be adopted, and multiagent design issues. Modeling is hard, since it deals with difficult problems such as planning, multicriteria decision, agents coordination, etc. The generated model represents, in a more generic way, the structure of decisions in a game and it is an essential input to the next stage.

In the implementation stage, the specific AI techniques (such as rule-based systems, supervised and unsupervised learning, etc.) are chosen, and the model generated previously is detailed and instantiated to fit the chosen AI technique. In the adjusting stage, tests are made and modifications are performed to satisfy the game requirements.

All this complexity, in particular for RTS, makes the development game AI a costly task. This often leads to a simplification of game AI, which may result in a low player experience.

Unfortunately, the current literature on game AI, particularly in AI for RTS, only describes the application of a AI technique to specific problems [Weber 2009][Weber 2011][Ívarsson 2005][Miles 2006][Cheng 2004]. Very little attention is given the modeling stage and it does not cover the whole game with all the problems, tasks, decisions, etc.

In this paper, in order to help the game developer, we introduce an original method for the first stage of modeling a game AI. This method will map in an easy way, the problems that require some AI. The method is decision-centered and enumerates a series of simple questions that the developer must answer to figure out the AI game problems.

This model emerged from a five-year effort of observing and coaching more than 250 students of an undergraduate discipline (Autonomous Agents and Multiagent Systems) taught in the Informatics Center of the Federal University of Pernambuco (CIn-UPFE). The student form teams to program the AI of NPCs of a strategy game. During the course, the AI techniques that can be used in the task as studied and hints are given on how to model the AI. At 3 moments of the course, a competition takes place among the teams, where the NPCs programmed by each team must fight against each other. The grade obtained by the students depends on their performance in the competition. In the last two years, we moved from a generic RTS game definition and a platform developed by our research team [Vieira Filho et al. 2007], to the StarCraft using the Broodwar platform [BWAPI 2012].

In the last version of the course, we have fully applied the method presented here to help the teams to define the AI. The results were clearly satisfactory as we discuss later.

The method developed was only used in RTS games so far. Despite this, it is possible to apply the method with few or no changes to other styles of games. For being one of the most complex game styles, we idealize that applying this method in other styles will have a result quite promising.

This paper is organized as follows: the second section we make a review about the others works in the area; following a section that defines the method that we used for modeling an AI of a RTS game. The fourth section we show the model that we build. In the fifth we discuss the results and the last section we finalizes this work.

2. Related Work

RTS games may have hundreds to thousands of agents that interact with each other, with the environment and with the player. The better intelligence those agent actions has better will be the user experience.

2.1 Multi-Agent Systems

Although the development of Multi-Agent Systems (MAS) currently being a technique quite widespread, it is still held, in the vast majority of times, without the aid of a methodology as a guide. One of the reasons is the inadequacy of MAS' development to current engineering techniques [Fisher 1997]. However, there are already some approaches that seek to bring software development techniques for the development of SMA. Techniques such as: **(i)** Tropos [Bresciani 2003]; **(ii)** MaSE [Deloach 2006]; and **(iii)** SADAAM [Clynch 2007].

Tropos [Bresciani 2003] methodology guide the whole software development process using the concept of agents still the beginning. However, this approach is very rigid as some old software engineering techniques, making the multi agent system development very static.

While the approach of methodology MaSE [Deloach 2006] seeks to facilitate the development of the system by specifying a set of formal documents that will guide the developer during the remainder of the development cycle. However, it remains a very rigid methodology, contrary to some newer techniques of software engineering, such as the agile techniques. Where can we find contrary to excessive generation of technical documentation.

In the case of SADAAM [Clynch 2007] is used an approach based on agile methodologies, bringing together the worlds of some more conservative methods with the idea of flexibility introduced by Agile methodologies. The features incorporated by the agile methodologies regarding the ability to make development less costly the changes that may occur during the lifetime of the process.

In the case of the methodology proposed in this paper, we address a more agile process, allowing greater flexibility. Added to a more didactic and practical system of problems' identification that a multi-agent system will face.

2.2 RTS Modeling

Weber [Weber 2011] in his work entitled "Building Human-Level AI for real-time Strategy Games", reports the complexity in designing an intelligent system that controls the AI of a RTS game. In addition, Weber suggests the use of a hierarchy decisions. This choice, according to the author, makes it easier for a given level of the hierarchy to generate an activity that can be consumed by the correct management level. Who also follows this same idea of creating a hierarchy of decisions is Ívarsson [Ívarsson 2005].

Harmon, [Harmon 2002] used an economy based approach to Goal-Directed Reasoning which deals with the investment problem during an RTS game making analogy with the marginal utility economics' model. The marginal utility is used to model a system which adapts to dynamic reality of a RTS game environment.

In another work, Dahlbom [Dahlbom 2004] models an AI which aims to minimize the adaptation time of a player in a RTS game through dynamic scripting. Goal-based AI establishes relations between rules and goals, weight-adjustment, and not only takes into account the effective result of the rules but also considers the ability level of the enemy player and prevents the AI to learn in case of a unskilled enemy.

Churchill [Churchill 2011] proposes a model that seeks to optimize the order in which the construction of structures in a game of StarCraft is carried out. This approach is well defined, but is limited to a very specific point in the game, the construction of new buildings. In addition, as is focused on only one case of game, focuses on much in the way of implementing the proposed model, not abstracting the idea to other situations and/or cases of an RTS.

In his dissertation "AI for real-time strategy games", Walther [Walther 2006] describes many details of what is present in an environment of an RTS game. Among the techniques that are used in the development of RTS AI, Walther cites the idea of using a hierarchical modeling, where we would have the behaviors of the game basically divided into three levels: (i) low-level behavior, which deals directly with what each unit will be doing every moment; (ii) medium-level behavior, that would be the Group of control units; and (iii) the high-level control, which would be related to the General control of the game, which would encompass issues such as as choices of strategies and town construction. However, Walther did not formalize the architecture of control of the AI of the game, being then one more work that is focused on how to make AI

effective and not on how the architecture that would d do that could be designed.

Another work that uses the idea of a hierarchical architecture for AI of an RTS is the work of Safadi [Safadi 2011]. However, Safadi deals with the actions differently from most works. He defines that the highest level of the hierarchy would request a high-level activity (e.g. create a group of marines). In order to accomplish such high-level activities, each high-level generates some middle-level activities (create a barrack) that can, in their turn, generate several other low-level activities (searching resources). However, it's just another job that has his contribution focused only on the development of AI and not in the definition of a template that can be reused by other surveys.

In the next section we will discuss the process that we use for the idealization of a model capable of representing the AI of a RTS game.

3. Problems Mapping Method

All this difficulty motivated us to propose a methodology that helps with configuring the development of an AI for a RTS game. This methodology was put into use in a case study over the course of an undergraduate discipline called Autonomous Agents (AA), which was taught in semester 2012.1, in the Center of Informatics of the Federal University of Pernambuco (CIn-UFPE).

During this semester, we did some reflection exercises with the students who have a better view on defining a complex AI system. These exercises allowed us, for example, to identify which the decisions were taken by a player during a game of an RTS game. Furthermore, they allowed us to examine, in the case of an AI, which part of the system would take each decision or at what time.

In order to leverage the RTS AI researches, this paper models a Multi Agent System Design Process which provides to the researchers of this area a method that guides their studies and helps in understanding the whole RTS AI problem.

3.1 Autonomous Agents

The discipline of AA is currently once per school year. When it was last offered (2012.1), 50 students attended. As an incentive to make the subject a bit more tangible to students, we put in place a MAS design process. This method helped the definition of an intelligent mechanism able to execute the control of an RTS game.

This method was used by all students, in some discursive classes. From the application of this method, we reached the goal of an organizational structure capable of representing architectural decisions that the AI of a RTS game would make. As their final project discipline, the class was divided into five teams, each one responsible for developing their own AIs for control of the commercial game StarCraft-BroodWar, using BWAPI [BWAPI 2012]. This API allows developers to write a C code through which they can control the actions that occur in the game.

After the end of the course, we received positive feedback from the students about the appliance of the method. In a written exam, students demonstrated a greater theoretical knowledge on the subject and in the development of the project; it was possible to find the implementation of some concepts seen in the room.

3.2 Method

For the discipline of AA we used a method to assist the definition of AI in a RTS game. This method consists of several steps displayed in the listing below. In this listing, we see the main stages for which teams built their models.

1. List the decisions that must be made – *What should be done?*
2. Define the variables for each decision
3. Identify who should take each decision – *Who should done?*
4. Identify relationships and priorities for each trigger actions – *When decision should be taken?*
5. Decide *how to take each decision*

The first step is characterized by the questioning of "*what should be done*". After listing these activities other questions will guide the assembly of the rest of the model. However, before one has any other questions it is important to define which variables are taken into account for the decision to be taken. The definition of these variables will facilitate the future implementation, but also facilitate the analysis of the next process questions.

According to [Safadi 2011][Ívarsson 2005][Weber 2011], the better representation of a game AI would be hierarchical decisions. Hence, we would have in the next step the identification of which activities are being performed at each level of the hierarchy. To make this possible, we have posed a new question, that is: "*who should get it done?*". The answer to this question will tell us: **(i)** if this is a strategic decision, which is the highest level, like a general in an army; **(ii)** a tactical decision, that will be like a Commander of an army grouping; while the third answer **(iii)** would be an operational decision, which technically would be a decision at the level of a unit (soldier of the army). This step is useful in defining the environment variables that must be taken into account in making the decision, because these variables will determine the degree of knowledge about the environment that every decision must have.

Another issue that should be decided is when each decision will be made. This would be the third question of the method, namely "*when the decision should be taken*". The answers of this question, in relation to each decision listed earlier, will begin to give us an idea of the chronological execution of actions. What we defined in this step is the relationship between decisions that must be taken, **(i)** which comes first; **(ii)** which a decision requires external events as trigger; **(iii)** when a decision is part (sub decision) of another decision; and other temporal relations associated with these decisions. This will allow us to draw the first sketches of an intelligent system for AI of a complex RTS.

The next question asked, is basically what is already massively studied in other works, and relates to how to implement the decision of each one of the decisions listed. This step is not the focus of this work. Thus, we do not explore the subject in more depth.

After a semester, we had built a model that represents the AI of a RTS game. This model was the result of the application of the method will be discussed in more detail in the next section.

4. Model

Several studies have been conducted on RTS games, but each ones seeks the development of their AI in an arbitrary way. The purpose of this model is to facilitate the development of a new AI for a RTS games. With the template at hand, the developer will be able to better visualize the activities that AI must perform and thus organize in a better way the intelligent system architecture.

As main result from the application of this method in the discipline of AA was a model that can represents the decisions that occur in an RTS game. This model will allow the development of an intelligent system that will control the RTS game.

4.1 Decisions

Initially we can list the decisions that were identified. The list of 50 decisions can be seeing below.

1. Analyze defensive system.
2. Identify suitable points to enemy attack.
3. Need to strengthen defense?
4. There is unused in defense military unit that can be recruited?
5. Needs to build a new structure?
6. Needs to produce a new unity?
7. Have resource?
8. Which structure to build?
9. Needs to build a new command center?
10. Where to put the new structure?
11. Allocate workers.
12. Build the structure.
13. Some structure needs to be repaired?
14. Some army cavalry needs to be repaired?
15. Which unity to produce?
16. Which support structure will produce the unity?
17. Where to put the unity?
18. Analyze resource.
19. Need more resource?
20. Need more worker to get resources?
21. Which resource source to collect?
22. Set activity to the worker.
23. Need a new resource source?
24. Which resource source to acquire?
25. Command attack to attach the resource source.
26. Identify which new technology to produce.
27. Some unity needs medical care?
28. Allocate doctor to heal the unity.
29. Command doctor to heal.
30. Needs to expand the base?
31. Commands the defense system to expand.
32. Identify the moment to attack.
33. Which enemy base attack?
34. Which enemy defensive system is worse?
35. Which enemy structures are more important?
36. Define attack strategy.
37. What will be the composition of the attack?
38. When send backup army?
39. Identify difficult to destroy an enemy structure.
40. Identify which enemy unity to attack.
41. Who attack?
42. Make some enemy recognition?
43. Needs to abort the attack?
44. Command retreat.
45. Identify strategic points in the map.
46. Analyze the possibility to take control of the strategic point.
47. Command attack to take control of the strategic point.
48. Need some recognition patrol?
49. Allocated some unities to make the recognition patrol.
50. Identify the path of the recognition patrol.

Having this list, we can make the second question, "*who should done?"*, which will give us an idea of where in the intelligent system the decision will be make. The answer will tell us whether the decision will be made at a high level, in a centralized way, by the core of the system that controls the AI of the game; otherwise from the agent itself, using a distributed way happening a low level decision; or if we will have a new agent between these two, which will be responsible for decisions on the medium level, using an hybrid system.

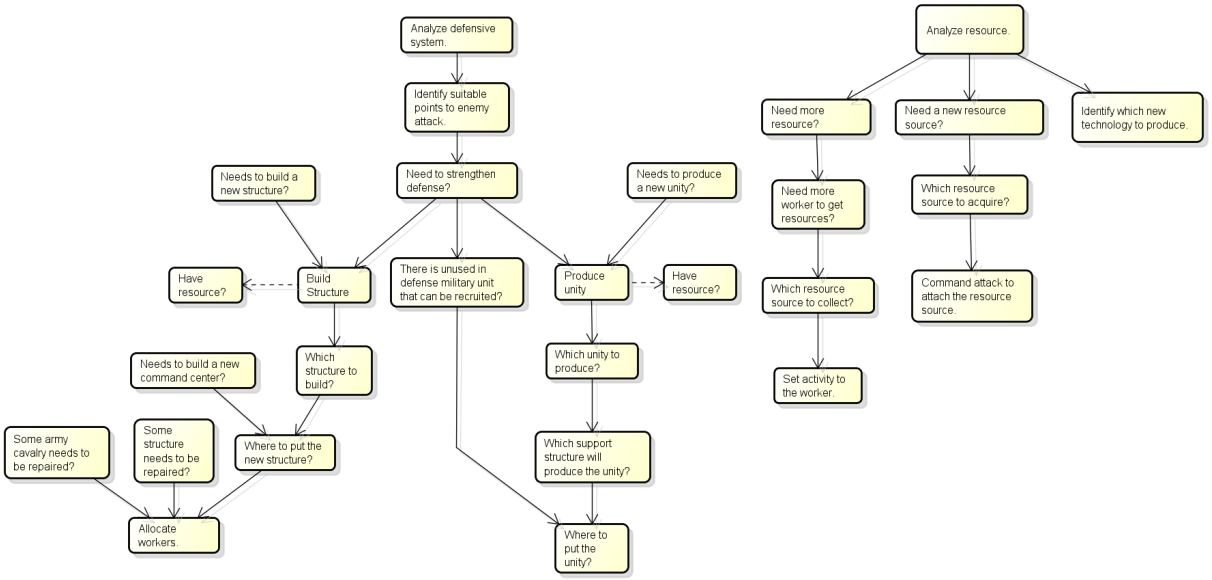

Figure 1 - The relationship of the AI decisions in a RTS game

For example, a decision of "need more worker to get resources" have to be done by the high level system, which will be able to take this decision without the participation of any other part of the system. Otherwise the decision "who attack", make a unit to attack an enemy, this decision have to be made by the agent itself without coordination. After this, the next step is to verify some relationships about each one of that decisions. This will facilitate to make a better analysis about each part of the system. The result of this analysis is a diagram that can be seen in Figure 1 and Figure 2.

Following the process previously seen, each of these decisions will be expanded into several other decisions. To help understanding, the examples used in this section will be in relation to a small part of the intelligent decision system for RTS. In this way, we will only deal with the case of how to perform base defense of his army.

We can see that in the Figure 3, this diagram presents not have only the activities, but also the representation of the main variables that can influence the decision-making. In addition, we also have the relationship between decisions, watching so that a single decision can have a list of chained decisions. Also in this same figure, we can see the insertion and the representation of what we consider triggers for the intelligent system to analyze a given decision-making.

Thus, this template already allows a broad view of the sequence of activities that an AI for a game RTS should accomplish. However, we still see some complex decisions, with a certain level of depth (levels

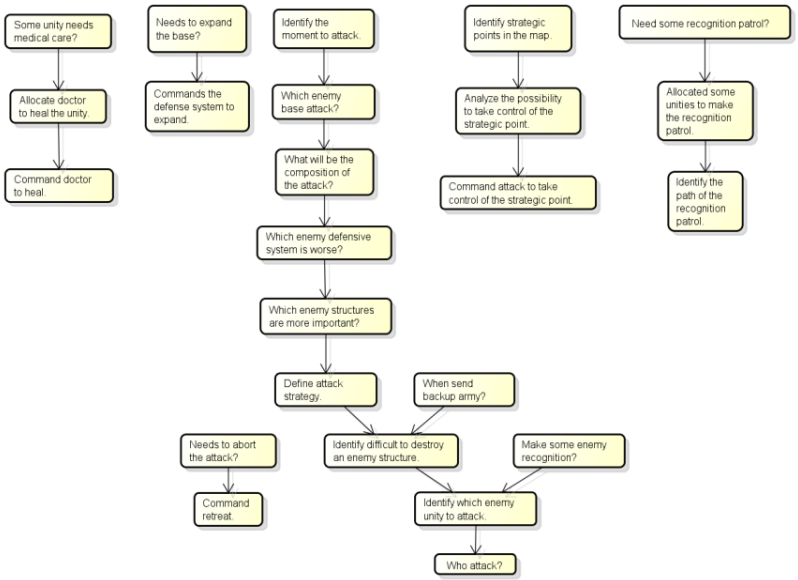

Figure 2 - (Cont.) The relationship of the AI decisions in a RTS game

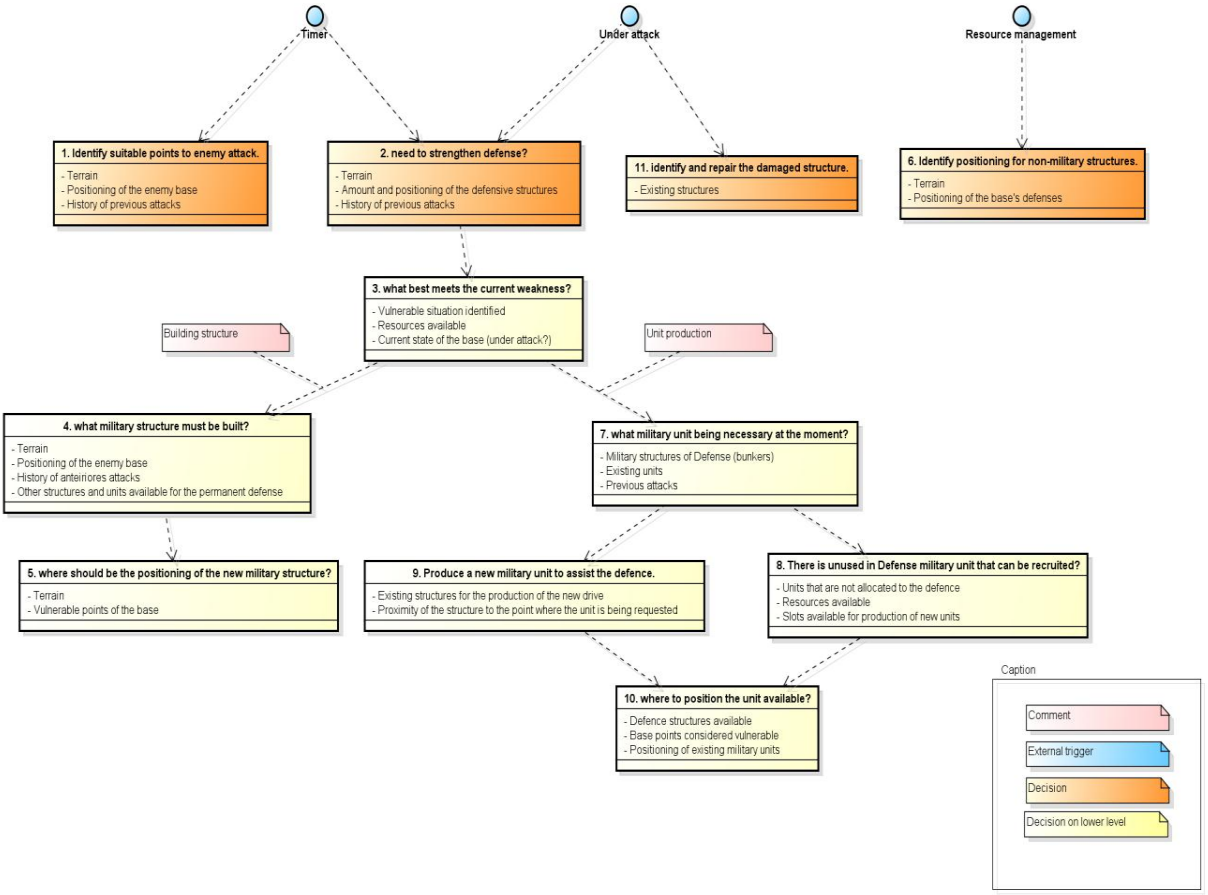

Figure 3 - Detailed defense activities with environment variables and decisions triggers

of nested decisions). All of this just confirms the complexity involved in the environment of an RTS game.

The solution was the creation of multiple state machines, where we would have one for each module identified.

4.2 Machine State

This model is able to represent the decisions in a god manner, including their relationships and when each one should be initiated. However, this model can still leave margins to be misinterpreted. In this way, we refine this model seeking a manner that would allow a better analysis by the developer of the AI system. For this new modeling, we use a state machine, because it removes the possible ambiguities that could exist.

In Figure 4 we can see the machine state that was generated for the defense case. This machine state will manage the defense of the base, checking when will be necessary to repair or build a new defense structure or recruiting a new unit to compose the defense.

A more complex case is the command center where we can see in the Figure 5. This machine state is responsible for manage the resources and acquisitions for new resources source. Beyond this, this machine state manages the situations of new structures, new unities and the research for new technologies.

Those machine states were compiled from the results obtained in the AA classes. For simplify the method, we just present some of the cases of an RTS game AI.

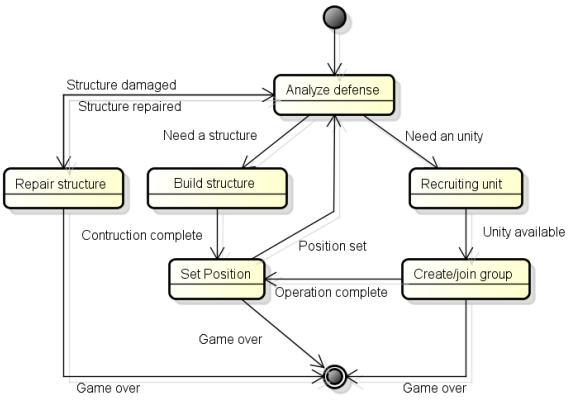

Figure 4 - Machine state of the defense AI

.

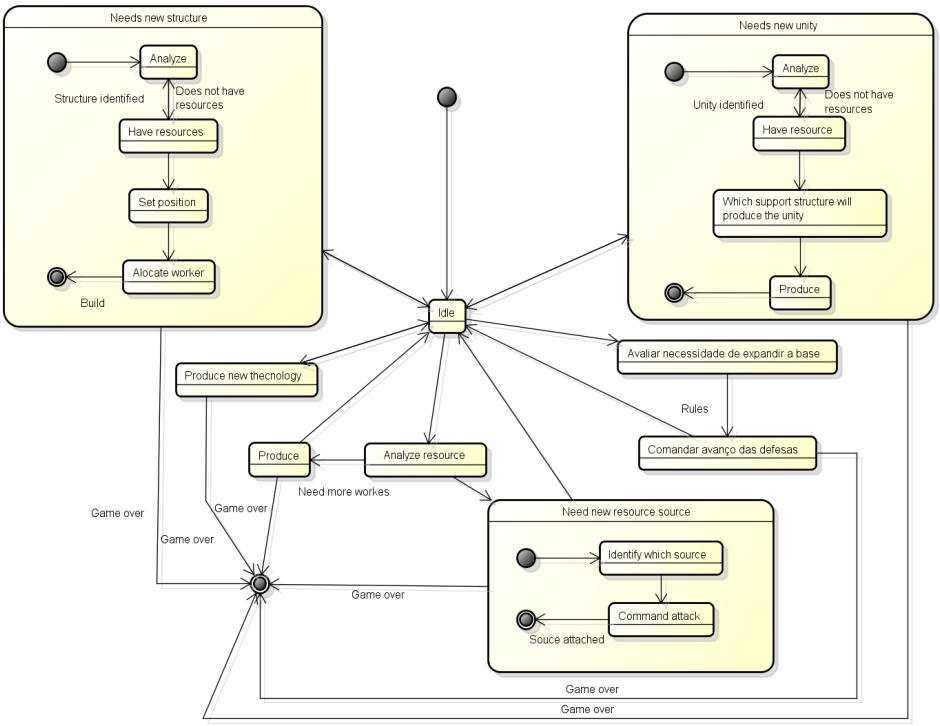

Figure 5 - A machine state that will represent the command center

5. Results

We stated to propose an AI programming for RTS as a competition in our course in 2006. Last year, as the result of observation and coaching, we finally arrived to the first mature version of a method presented in this paper

This year, 2012, we decided to explicitly teach the method to the 50 students we had in the course. We spent 15 min explaining the method and a couple of hours starting the modeling processing with them according to the method.

The results are beyond our best expectations. First, the NPC have clearly more intelligent behavior with respect to the ones from the previous version of the course. The majority of matches played by the 2012 teams against the previous ones corroborated our first good impressions, since they were won by the 2012's.

Second, the students' results in the written test were also better than the previous versions of the course. It is possible to compare results because we apply the same written test to all versions of this course. The questions may change a little syntactically, but they are the semantically equivalent. The questions, which are given in the very first day of the course, ask the students to explain how they implemented their agents (NPCs) using the concepts and terminology discussed during the classes. We noticed that their explanation were far more structured and clear after the application of the method explained in this paper.

6. Conclusion

This paper introduces preliminary results of a method that help game developers to structure and map the AI for a RTS games. The preliminary and practical evaluation of the method is encouraging when applied to help students to develop the AI of NPCs of Starcraft, a well-known and typical RTS game.

For future work, we will apply this method for other games stiles and beyond that, for other complex multi-agent systems. We expect that this method will be applied in those cases with little or no modification.

This paper also presents, as a side-effect contribution, a partial model of game AI for RTS. The description of the entire model is not in the scope of the paper indeed.

The method emerged from an academic course and we do not have the opportunity to test in other environment beyond the academic. So, this is a case study for the future, adding a further evaluation of the method by performing experiments with a control group.

As seen, there are a few methods that make it easy to develop MAS. However, the authors are not aware of a method that helps in the modeling of an intelligent

system designed for gaming. In this way, although we have only preliminary results, the proposed method has reached quite promising results.

References

BRESCIANI, P., GIORGINI, P., GIUNCHIGLIA, F., MYLOPOULOS, J., & PERINI, A., 2003. Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems.* May 2004, Volume 8, Issue 3, pp 203-236.

BOURG, D. M., & SEEMAN, G., 2004. AI for Game Developers. *Environments* (p. 400). O'Reilly.

BWAPI, 2012. An API for interacting with Starcraft: Broodwar. *Retrieved from http://code.google.com/p/bwapi/* in august 03th of 2012.

CHENG, D. C., & THAWONMAS, R., 2004. Case-based plan recognition for real-time strategy games. *Proceedings of the Fifth Game-On.*

CHURCHILL, D., & BURO, M., 2011. Build Order Optimization in StarCraft. *Seventh Artificial Intelligence and Interactive Digital*, 14-19.

CLYNCH, N. & COLLIER, R. W., 2007. SADAAM: Software Agent Development An Agile Methodology. *LADS/Durham Agents.*

DAHLBOM, A., 2004. An adaptive AI for real-time strategy games. *Institutionen för kommunikation och information* {p. 74}.

DELOACH, S. A., 2006. Engineering Organization-based Multiagent Systems. *LNCS.* 2006, pp 109-125.

FISHER, M., MÜLLER, J., SCHROEDER, M., WAGNER, G. & STANIFORD G., 1997. Methodological Foundations for Agent-Based Systems. *Knowledge Engineering Review.* September, 1997, Volume 12, pp 323-329.

HARMON, V., 2002. An Economic Approach to Goal-Directed Reasoning in an RTS. *Charles River Media.*

ÍVARSSON, Ó., 2005. Improved Combat Tactics of AI Agents in Real-Time Strategy Games Using Qualitative Spatial Reasoning. *Hogskolan Skovde.*

MILES, C., & LOUIS, S. J., 2006. Towards the co-evolution of influence map tree based strategy game players. *Computational Intelligence and Games,* 2006 IEEE Symposium on (pp. 75–82). IEEE.

ROLLINGS, A., & MORRIS, D., 2003. Game Architecture and Design - A New Edition. *New Riders Games.* p. 960.

Safadi, F., Fonteneau, R., & Ernst, D., 2011. Artificial Intelligence Design for Real-time Strategy Games. *25th Annual Conference on Neural Information Processing Systems (NIPS 2011)*, 1-6.

VIEIRA FILHO, V., SIEBRA, C., MOURA JÚNIOR, J.C., WEBER, R. & RAMALHO, G., 2007. RTSCup Project: Challenges and Benchmarks. *In: Anais do VI Simpósio Brasileiro de Jogos e Entretenimento Digital, SBGames 2007,* (pp. 163-179). São Leopoldo: Sociedade Brasileira e Computação. ISBN: 857669154-X

WALTHER, A., KIM STEENSTRUP PEDERSEN, & LOOG, M., 2006. AI for real-time strategy games. *Learning. IT-University of Copenhagen Design.*

WEBER, B. G., & MATEAS, M., 2011. Building Human-Level AI for Real-Time Strategy Games. *AAAI Fall Symposium on Advances in Cognitive Systems.*

WEBER, B. G., & MATEAS, M., 2009. A data mining approach to strategy prediction. *2009 IEEE Symposium on Computational Intelligence and Games*, 140-147. Ieee. doi:10.1109/CIG.2009.5286483