# STRATEGO EXPERT SYSTEM SHELL

**Casper Treijtel** and **Leon Rothkrantz**
Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4 2628 CD Delft University of Technology
E-mail: L.J.M.Rothkrantz@cs.tudelft.nl

## KEYWORDS

Games, A.I., Multi-agent, Expert systems, Stratego

## ABSTRACT

The field of multi-agent systems is an active area of research. One of the possible applications of a multi-agent system is the use of distributed techniques for problem solving. Instead of approaching the problem from a central point of view, a multi-agent system can impose a new mode of reasoning by breaking the problem down in a totally different way.

In this paper we investigate a distributed approach to playing Stratego. Computational agents that each have their own field of perception, evaluation and behavior represent the individual pieces of the Stratego army.

A first prototype of a framework has been developed that consists of a simulation environment for the agents and an implementation of the agent's evaluation function. The agents have a rule engine that generates behavior that is a resultant of the environment in which they live.

## INTRODUCTION

This paper describes a first attempt to play the Stratego game with multiple agents. The Stratego game is a board game where two players battle each other with their armies of pieces. The object of the game is to capture the enemy flag, by moving pieces towards the enemy and trying to capture the enemy pieces. An interesting property of the game is that the information the players have is incomplete, because the identity of the opponent's pieces is concealed until exposed by battles between pieces.

Our motivations for using the multiple agent approach are as follows. When we consider a human society from a central point of view we see that it is a very complex system. A possible attempt to understand the complex behavior of a human society is to consider it as a system that is made up of individuals that have their own characteristics, behavior patterns and interactions with each other. It is the sum of all the local actions and interactions that constitutes the overall behavior of the society. This investigation is an attempt to support this hypothesis by considering the Stratego game. Specifically we want to investigate whether a distributed way of playing this game will provide us with a means to break down the complexity of playing it.

Our work is based on ideas of multiple agents as described by J. Ferber (Ferber 1999) and intelligent agents as developed by P.Maes (Maes 1995) and L.Steels (Steels 1997).

## DESIGN

In designing the agents we want to make use of the fact that each piece in the Stratego army has a certain dedicated role. These roles originate from their specific ranks and the rules of the Stratego game. All pieces have secondary goals as well of which possibly the most important one is to stay alive. We propose to define some degrees of freedom in our model of the agent that will allow us to experiment with different types of agents in the Stratego army. Specifically we define for each agent:

- The agent's perception range. Depending on the agent's role in the army the perception will be a diamond of range one to five, or an n x n square of fields. Important pieces will have wider perceptions.
- The agent's 'reactive' behavior. For every agent we define four elementary behaviors that are executed following a reaction in various situations. These behaviors are attack, flee, random walk, and stay and do nothing.
- The agent's 'cognitive' abilities, for example evaluate situation, compute optimal next move, form hypotheses, and make plans.

In our design emotion is modeled as follows. Emotions are related to parameter settings regarding the agent's perception and behavior. For example, if an agent gets upset, afraid or stressed we shrink his field of perception (tunnel view). And if the agent is angry we increase the possibility to attack (McCauley 1998; Scheutz 2000).

We designed two levels of communication among agents. One is communication by means of a blackboard that can be written to and read from by every agent. The blackboard is a container of all information of the board situation that is available. This way all agents can rely on the fact that their field of perception is in accordance with the current board-situation. The blackboard contains strictly information about the board status.

Additionally the agents can use an asynchronous message-passing structure. Agents can send and receive messages to each other containing information about the Stratego battlefield. The communication structure allows sending messages to all other agents, sending messages to agents of a certain rank or sending messages to specific agents. The content of messages can either be known facts, hypotheses or requests.

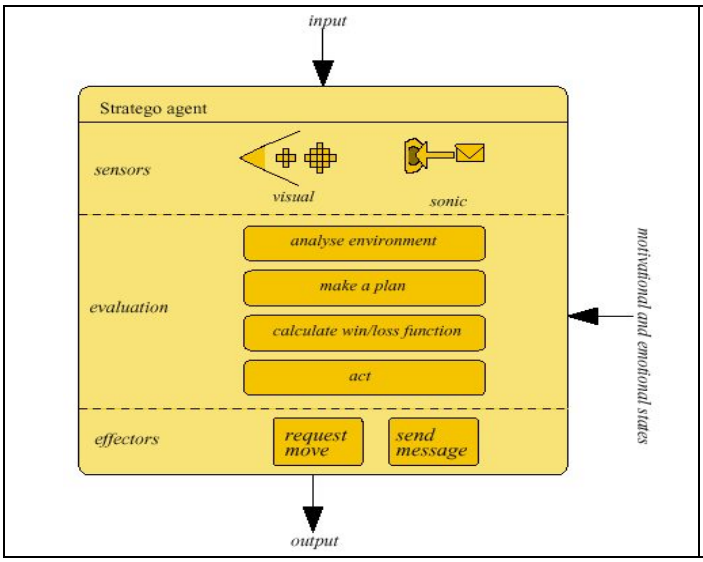


Figure 1: Traditional approach to modeling an Stratego agent based on a functional composition of modules in the evaluation layer.

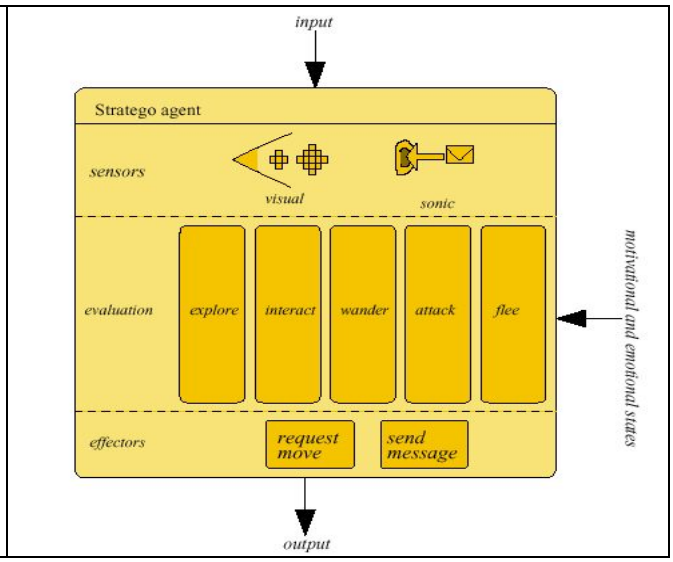


Figure 2: Behavior-based model of the Stratego agent with separate behavior modules in the evaluation layer.

Because only one piece can move at a time, a mechanism was designed that decides which agent is allowed to move. The decision rule was based on scores, where each agent evaluates its current situation and assigns scores to preferences of moving. A higher score will indicate a stronger desire to move and the agent with the highest score will be allowed to move.

## ARCHITECTURE OF THE STRATEGO AGENT

For our Stratego agent we defined a three-layered architecture, with a sensor, evaluation and effector layer. These layers relate sensor inputs to actuator outputs. The actual relation between percepts and actions takes place in the evaluation layer. There are various possibilities for filling in the evaluation layer. We discuss the traditional and the behavior-based approach designed by R. Brooks (Brooks 1986).

The traditional approaches to model cognitive systems are based upon a strict functional decomposition of modules. These approaches result in so-called sense-model-plan-act frameworks. The cognitive system contains a number of modules that are built on top of each other, each performing a dedicated function as a part of the system.

One characteristic of these types of frameworks is that every module has a specific function that uses input from the module before it. When applied to the Stratego agent, the traditional framework takes the form as indicated in Figure 1. The three layers, (sensors, evaluation and effectors) are influenced by the motivational and emotional states that the agent undergoes.

The behavior-based approach has the advantage that new modules with new behaviors can be added to the system quite easily. Also, the architecture allows for a combination of modules that may be based on each other or that may be conflicting among each other. It is imaginable that some goals of Stratego agents may very well be conflicting. The architecture of the behavior-based approach seems to be very appropriate for our notion of the Stratego agent, in the sense that for each goal we are able to add a separate behavior module. The three layers are influenced by the motivational and emotional states that the agent undergoes.

As is the case in the subsumption architecture, these modules operate in a considerable autonomous way. The modules shown in Figure 2 are some behaviors that apply to a piece in a Stratego environment. Depending on the situation at hand, one of the behaviors has the overhand and dictates the overall behavior of the agent.

## KNOWLEDGE OF THE AGENTS

Since the agents represent pieces of the Stratego army, we want them to express behavior that can be seen as 'rational' from their point of view. In other words, we want them to express behavior that will make the agents successful in achieving their goals. Our approach is based on a rule-set that explicitly defines what to do for a number of situations.

For each of the Stratego agents we have defined a set of rules that specify the behavior, according to the current situation of the agent. We call these rule-sets preference rules, since they indicate preferences to exhibit behavior rather than performing explicit actions. The use of preferences instead of actions in the rules arises from the desire to allow separate behaviors to be activated simultaneously.

We will give some examples of preference rules of the set of 29 preference rules for the "minor"-agent:

**Rule 1:** This rule will fire the preference "attack" when the following conditions are met:

- Enemy bombs captured
- I have moved
- My rank revealed
- Enemy with unknown rank present at distance 1

**Rule 13:** This rule will force the preference "flee" when the following conditions are met:

- I have moved
- My rank revealed
- NOT enemy bombs captured
- Enemy with unknown rank present at distance 1

**Rule 22:** This rule will fire the preference "stay" when the following conditions are met:

- NOT I have moved
- NOT my rank revealed
- NOT enemy bombs captured
- Enemy with higher rank present at distance 1

**Rule 27:** This rule will fire the preference "attack" if an only if an enemy bomb has been spotted at distance 1.

Upon each move, all agents evaluate their situation and express their desire to act or not. Because of the fact that only one agent can and has to move at a time, one agent has to be selected. This is done according to a weighted function that takes into account all desires of agents. The agent's rule engine has been implemented using the notion of separate behavior patterns that conform to the behavior-based model (Figure 2). The agent's behavior can be explained as being a resultant of all separate behaviors. The agents show emergent behavior that is caused by the sum of all separate behaviors.

A great advantage of this type of emergent behavior is that the agent comes somewhat closer to our notion of an autonomous system. The agent's perception, goals, motivations, etc. all influence the agent's actions. This means that we can define different types of rules for the agent that may harmonize or conflict with each other.

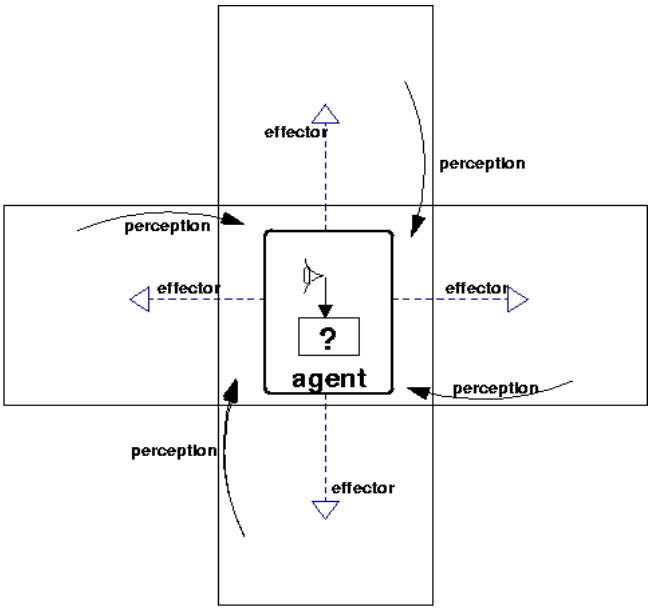


Figure 3: The environment in which the agent playing Stratego lives

## IMPLEMENTATION

In this section we will describe an implementation for a prototype called Stratesys, as an acronym of the words Stratego expert system shell. The implementation has been done using the object-oriented programming language Java2. In the current version of the Stratesys we have implemented an agent type that is based on production systems. For the communication among agents and the agent rule-engine we have used the JavaSpaces Technology and an expert system shell called Jess, respectively.

### The simulation

According to Russel & Norvig (Russel et all. 1995), an agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors (see Figure 3). All agents have the three layers sensors, evaluation and effectors. See Figure 4 for a schematic view of the agent. Here we can see each layer containing the agent's internals. It also shows the objects it is related to in its environment.

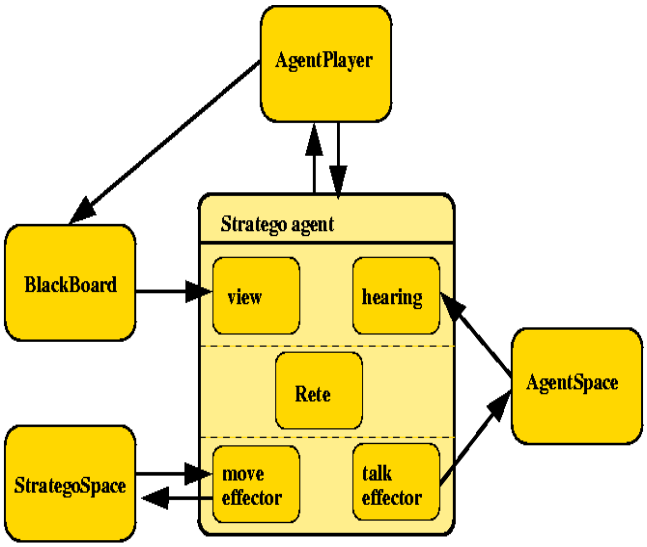


Figure 4: Schematic view of the agent's implementation

The Agent Player functions as a representative of the Stratego army formed by agents. It is responsible for creating all agents upon start-up, initializing them and positioning them on the Stratego board. Also, the Agent Player is responsible for maintaining information on the Blackboard (Cavazza 2000). This is an object that continuously reflects the actual situation on the Stratego board, the way the Agent Player sees it. In other words the Agent Player keeps positions of all pieces and where possible fills in missing information concerning enemy ranks.

The Agent Space is the agent's interface to communicate with its fellow agents (a JavaSpace-service). It is read from by the Hearing object and sent to by the Talk effector. The View object provides the agents with visual perception. It is actually an accurate copy of a small part of the Blackboard. It continuously checks for recent changes on the Blackboard, and updates itself whenever necessary.

The Stratego Space is the communication medium for the Client and the Server. The agent's lifecycle can be viewed as a number of states and transitions. The most important state in the cycle is the Evaluate state. Here, the Rete algorithm is applied using the percepts that have been received. If the Evaluation leads to an action, it will cause a transition to the Sleep state. In the Move state a piece can do an actual move. From the Move state there are two possible transitions to other states. When a move to an empty square was done the agent perceives some changes in

its environment and evaluates them. The other possibility is a battle with an enemy piece. In the Battle state the agent either wins and notifies all fellow agents of the capture, or the agent looses and notifies its death.

## The Client-Server model

Since we wanted to be able to play human versus human games, we have created two programs that implement a Client-Server model. The Client is the main Stratesys program. The Server runs in the background, continuously listening for Clients to connect. See Figure 5 for a schematic view of the Client and the Server.

The communication between the Clients en the Server has been implemented by a Java Space-service called 'Stratego Service Space'. Using the space the Clients and the Server can exchange information by reading from and writing messages to the space.

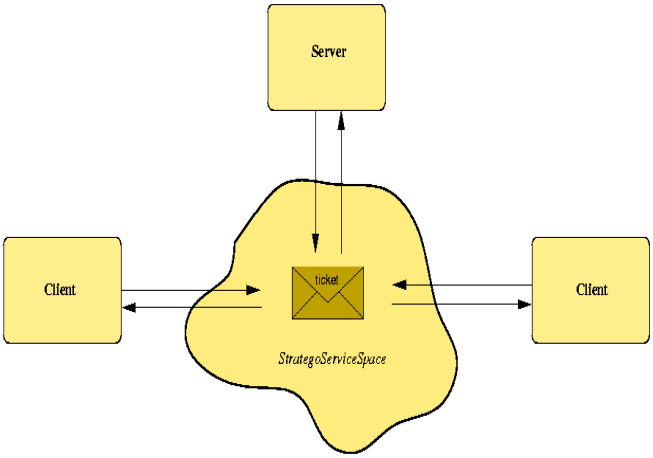


Figure 5: The Client-Server model

The Client is the main Stratesys program. At startup, a window is positioned on the screen with an empty Stratego board. The possible ways of playing the game are a human player playing against an agent army and two human players playing against each other.

At startup, the players will be registered with the Server. The human playing Stratego can position his army by clicking on the squares of the board. A pop-up menu will appear that will allow the player to choose a piece. When all pieces have been positioned, the players can begin to move their pieces. By using mouse-clicks on the squares of the board the human player can select pieces to move. For clarity concerning the situation on board we have chosen to implement the use of animation for each moving piece. When a correct move has been requested the board draws an animation of the moving piece from the initial position to its destination.

Depending on the type of game that is played, one or two Player objects will be created. Only in the human versus human mode will the Client create a one Player object. Naturally this implicates the necessity of another Client in the network. In the other modes of operation, only one Client is used which runs both Player objects. The Player object has both references to its own Pieces and to Enemy Pieces. The Enemy Pieces are actually only 'dummy' Pieces that are a visual representation of the actual enemy pieces. From the Player object to the Server and back are messages to register the player with the board. Messages from Player to Pieces concern position and move messages. The same applies to the messages sent from the Pieces to the Server and back. The Enemy Pieces however only receive messages from the Server and relay them to the Player object. This is because of the fact that these objects are only visual representations, as mentioned before.

Our implementation of the Server can accept two players wanting to play Stratego. These players can reside in one Client program or two. The latter case is only for human versus human games. After the game is over, the Server will wait for new requests for playing. The Client-Server communication consists of four phases. These are registration, positioning pieces, moving pieces and notifying a game over. For each of the phases we have defined specific messages, which we call, tickets.

Tickets are sent as requests and received as answers to that request. The idea behind the concept of a ticket is that a ticket gives a piece the right to position itself somewhere or move to a certain square.

Upon starting the game, the Client creates one or two Player objects, depending on the type of game that is played. The Players send a Registration Ticket to the Server to register. After sending the ticket, they will receive an answer with information about the registration (successful or not).

When two players have registered to the Server, they can position their pieces. For each piece to be positioned a Position Ticket is created and sent to the Server. The Server checks to see if the requested positions are valid, and send the tickets back with this information.

## EXAMPLE OF A TEST RUN

In this section we will consider two situations where the sergeant is in the environment as indicated in Figure 6. The sergeant sees an enemy piece with unknown rank (north square) and an enemy scout (northeast square). We will consider the case where the sergeant has already moved and its rank is known. The JESS output gives:

f-51 (enemy-known north east)
f-52 (enemy-unknown north)
f-54 (flee)
f-54 (update scores 0,–200,50,200,50)
f-55 (attack)
f-56 (update-scores 0,50,-50,-50,50)

Let us consider the computation of the scores (see Figure 7) in case that the sergeant has a desire to attack:

| | |
|---|---|
| Score for staying: | 0 |
| Score for moving forward: | -200 |
| Score for moving left: | 50 |
| Score for moving backward: | 200 |
| Score for moving right: | 50 |

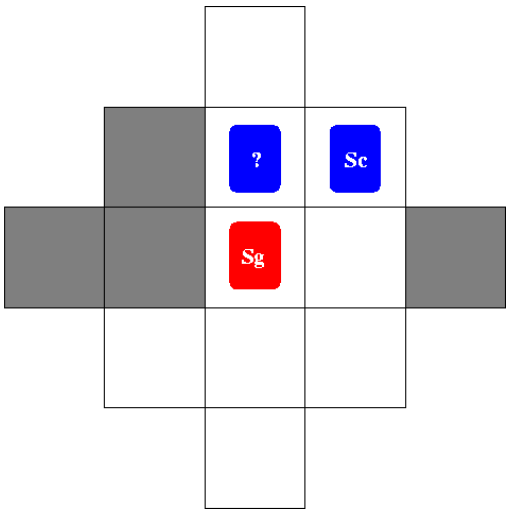


Figure 6: The sergeant's environment

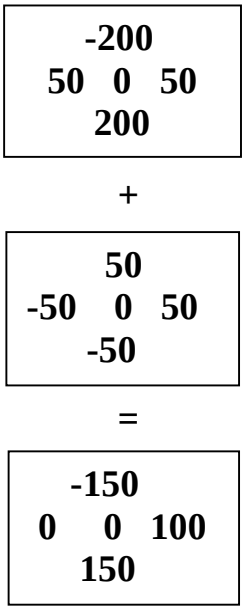


Figure 7: Computation of the scores

In the current implementation of the rule engine, the evaluation consists of a mapping from enemy location to a desire to move (for each direction) or to stay, expressed in scores. In the specific example, the sergeant may want to flee from the unknown enemy. But it also sees an enemy scout that can be beaten. Therefore in this particular case the sergeant's behavior will be a mixture of the desire to flee or to attack.

The scores indicated above express relative desires to go or to attack. Negative scores mean that the agent does not want to go in the corresponding direction. In the example the scores are a resultant of the behaviors to attack or to flee. The fleeing behavior is due to the enemy with unknown rank. Since the sergeant is a piece with a relative low rank, the score to move backward is largest and the sergeant will decide to move backward

## CONCLUSION

In this paper we have described a multi-agent approach for playing the game Stratego. This approach involves playing the Stratego game with multiple agents that each represents a piece in the Stratego army. The approach was based on the hypothesis that for some complex problems distributing techniques for solving them can result in more intuitive solutions. We assumed that the Stratego game could serve as an excellent playground for testing the hypothesis. Players have incomplete information on the board status and that results in the high complexity of the game.

We did not make an analysis of the game. We advocate using a corpus-based approach to build up a library of games, which can be used for studies and experiments about Stratego. The Client-Server model that has been implemented provides a framework from which several experiments can be run.

We have tested our prototype program Stratesys by letting the agents play against a human player. The experiments have resulted in some valuable ideas about our multi-agent approach. It proved that playing the game with multiple agents is an excellent approach to break down the complexity of the game.

## REFERENCES

Brooks, R.A. (1986). A robust layered control system for a mobile robot, *IEEE Journal of Robotics and Automation* RA-2:14-23

Cavazza, M, et al (2000), A real-time blackboard system for interpreting agent messages, *Proc. GAME-ON 2000*,49-55

Ferber, J. (1999). Multi-Agent Systems, *An Introduction to Distributed Artificial Intelligence*, Addison Wesley, England

Maes, P. (1995). Artificial life meets entertainment: Life like autonomous agents, *Communications of the ACM* 38, 11:108-114

McCauley, T.L. & Franklin, S. (1998). An architecture for emotion, *AAAI 1998 Fall Symposium "Emotional and Intelligent: The Tangled Knot of Cognition"*, AAAI Press.

Russel, S .& Norvig, P. (1995). *Artificial Intelligence-A modern Approach*, Englewood Cliffs, NJ: Prentice Hall

Scheutz, M. et al (2000) Emotional states and realistic agent behavior, *Proc. GAME-ON 2000*, 81-87

Steels, L (1997). A selection mechanism for autonomous behavior acquisition, *Robotics and Autonomous Systems* 20: 117-131