 UNIVERSITY OF CALGARY	Course Number: SENG 609.22	Course Name: Agent-based Software Engineering
	Session: Fall, 2004	Department: Electrical and Computer Engineering
		Document Type: Project Report

Contract Negotiation System
(for Professional Team Sports)

Scott Yu Tseng Su
SENG609.22
2004-10-31

I Table of Contents

I	Table of Contents	1
II	Abstract.....	2
III	Introduction	3
IV	Contract Negotiation Rules.....	3
V	Negotiation System Specification.....	5
V.1	Use Case #1: Player – Desktop PC.....	5
V.2	Use Case #2: Player – Networked Mobile Device.....	7
V.3	Use Case #3: Team Owner – Desktop PC	7
V.4	System Design/Architecture	8
V.5	Player Agents	14
V.6	Team Owner Agents	18
V.7	Contract Registrar	22
V.8	Auditor.....	23
V.9	Directory Service	25
V.10	Information Services.....	25
VI	Target Platforms	26

VII	Inter-Agent Messaging and Ontology Considerations	27
VIII	Possible Future Expansions	29
IX	Discussion.....	30
X	References	32

II Abstract

The design of an agent based contract negotiation system for professional team sports is presented. The system assumes a negotiation environment governed by a set of (perhaps artificial) rules where, in particular, a budget caps for each team is strictly enforced. Most agents in this system are in a competitive relationship with each other; a key feature of the system is that, in some inter-agent interactions, agents are permitted to provide false information to improve their position with respect to competing agents. This document is formatted somewhat as a project proposal.

III Introduction

The recent breakdown of contract negotiation in the National Hockey League, leading eventually to the current lock-out, highlights the need for a fairer and more efficient approach to contract negotiations amongst the team owners and players in professional team sports. This document proposes a set of simple rules to govern the contract negotiation process, and then presents the design specification of an agent based software system that implements these rules. The system will allow the contract negotiation process to be completed quickly and easily, with little need for user input (from either team owners or players) beyond the initial configuration of the software clients according to personal preferences.

IV Contract Negotiation Rules

1. **All teams share a “budget cap” that is strictly enforced.** One of the main points of contention in the current NHL contract dispute is whether a salary cap for players should be imposed. Players insist that their salary should directly reflect their skills and popularity, and that any artificial barrier to that is “unfair”; team owners, on the other hand, argue that without a salary cap, teams with smaller budgets are placed at too great a disadvantage – the more skilled players would all gravitate towards teams that can offer higher salaries. Without skilled players, the poorer teams’ game performances suffer, leading to lower ticket sells and, ultimately, an even smaller budget for the next year – a vicious cycle. A simple solution to this dilemma is to impose a cap, not on the players’ salaries, but on each

team's budget. In this way no individual player's salary would be limited (as a team's budget would be much larger than any single player's salary), yet the more successful teams would not possess overwhelming financial advantages over less fortunate teams in future years.

2. **Contracts may be offered/withdrawn and accepted/refused at any time.** Team owners and players are not obligated to initiate or complete negotiations at specific times. Of course, all contract negotiations must be completed before the start of a gaming season.
3. **Accepted contracts are registered with a central Contract Registrar, and become public knowledge.** Publishing all accepted contracts makes the task of enforcing each team's budget cap trivial. ("Under the table" deals are beyond the scope of this document)
4. **Accepted contracts may be voided by either party so long as a penalty is met.** It is unrealistic to expect players to team owners to remain bounded to a contract if a better prospect (say, a higher salary for a player, or a more skilled player for a team) becomes available before the start of a gaming season. So instead, each contract will specify a monetary penalty that must be paid before the contract may be legally voided.

The above four rules is sufficient to allow one to define a contract negotiation framework within which a multi-agent software system may be implemented. The following sections detail such a system.

V Negotiation System Specification

The negotiation system targets as its users both the team owners and the players. Broadly speaking, the function of the system is to facilitate the transfer and distribution of two resources: money (from the team owners to the players) and athletic skills (from the players to the team owners). Following are the use cases concerning these two types of users.

V.1 Use Case #1: Player – Desktop PC

Players may access the negotiation system through any PC so long as it has an internet connection. A software installation would be required unless the GUI front-end to the negotiation system is implemented as a web-based interface. The GUI will be designed with the novice user in mind, and should require little if any training; Fig. 1 shows one possible design for the GUI.

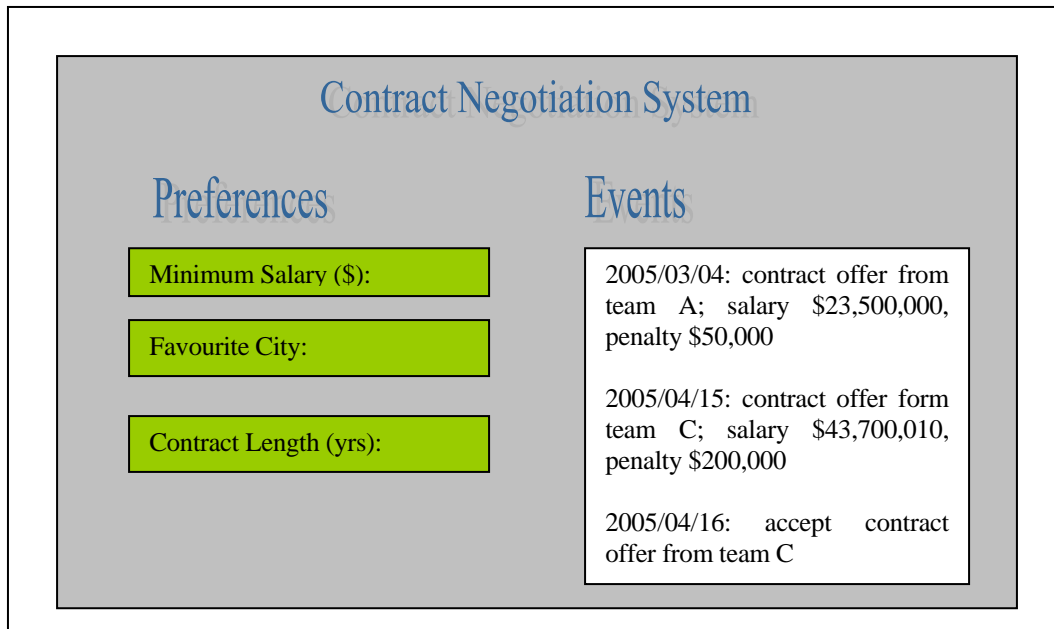


Figure 1 player GUI

If a player is accessing the system for the first time, he will be asked to input a series of personal preferences, such as minimum acceptable salary and preferred home city. These inputs will allow the system to cater to each individual player's personal requirements, which are expected to vary greatly – not all players will pursue monetary gains to the exclusion of all else. Once this initial configuration is done, the system will be ready for negotiations (i.e. ready to process contract offers) – no further input is required from the player. The player may re-access the system at any time to modify his personal preferences, or to monitor the current progress of the contract negotiation process that the system is conducting on the player's behalf. The GUI front-end may also be configured to provide notifications whenever an event (such as the arrival of a contract offer) occurs, or to request user confirmation before proceeding with certain actions (such as acceptance of a contract offer) – it is expected that initially users will not feel comfortable with the

complete automation of the negotiation process, and thus would prefer to manually intervene from time to time.

V.2 Use Case #2: Player – Networked Mobile Device

This is identical to Use Case #1 except for the GUI front-end, which will be executed from mobile devices with networking capabilities such as cell phones and PDAs.

V.3 Use Case #3: Team Owner – Desktop PC

Team owners will also access the contract negotiation system through a GUI front-end that executes on a desktop PC. The tasks that the system must perform on behalf of the team owners are much more complicated than those of the players, involving a large number of contract negotiations executing concurrently – a team owner may even wish to simultaneously negotiate with *all* available players. This increased complexity will be reflected in the GUI, and it is expected that some end user trainings will be required to become comfortable with the interface. The team owner interface may look something like Fig 2:

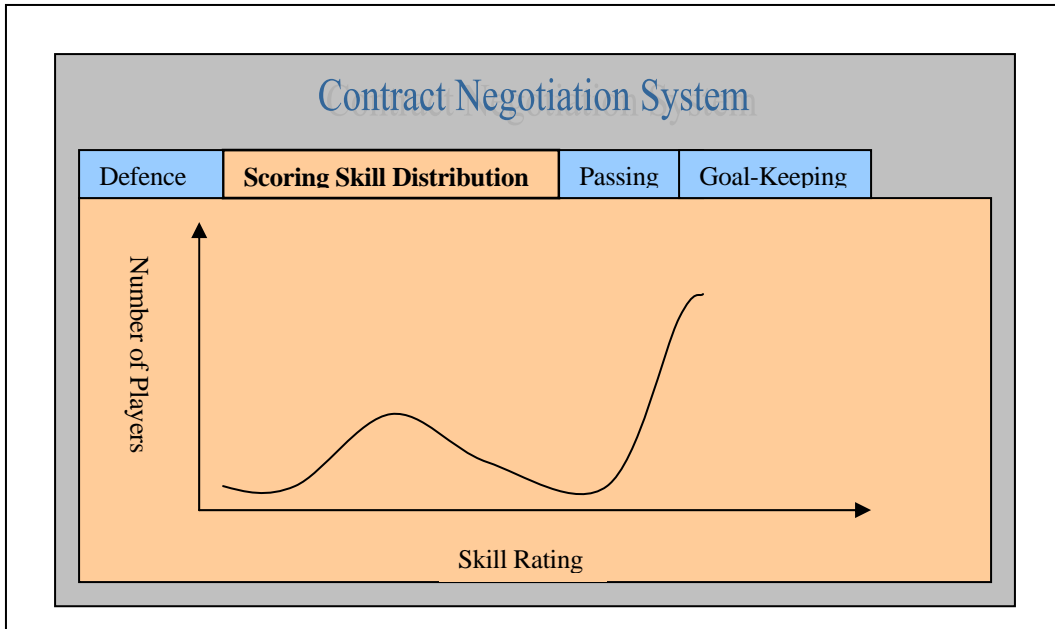


Figure 2 team owner GUI

As is the case for the players, the system is capable of conducting all necessary negotiations autonomously after initial configuration. The configuration process will be much more involved for the team owners, however.

V.4 System Design/Architecture

It should be clear that, in the majority of cases, the interests of system's users do not coincide: the players are competing against each other for a limited quantity of money; the team owners are competing against each other for a limited quantity of skills. The opposing interests of the users makes this system eminently suitable to an agent-based design: where a monolithic system will have to attempt very sophisticated multi-parameter optimization algorithms that takes into consideration the costs/benefits to every user, an agent-based system can simply implement an

agent to represent the interest of each user. Here the initial design of an agent based contract negotiation system is presented.

The system will be composed of the following four types of agents:

1. **Player** Agents
2. **Team Owner** Agents
3. Contract Management Services (**Contract Registrar** service and **Auditor** service)
4. Information Services (includes a **Directory** service, amongst others)

Fig. 3 illustrates the interaction model of these agents.

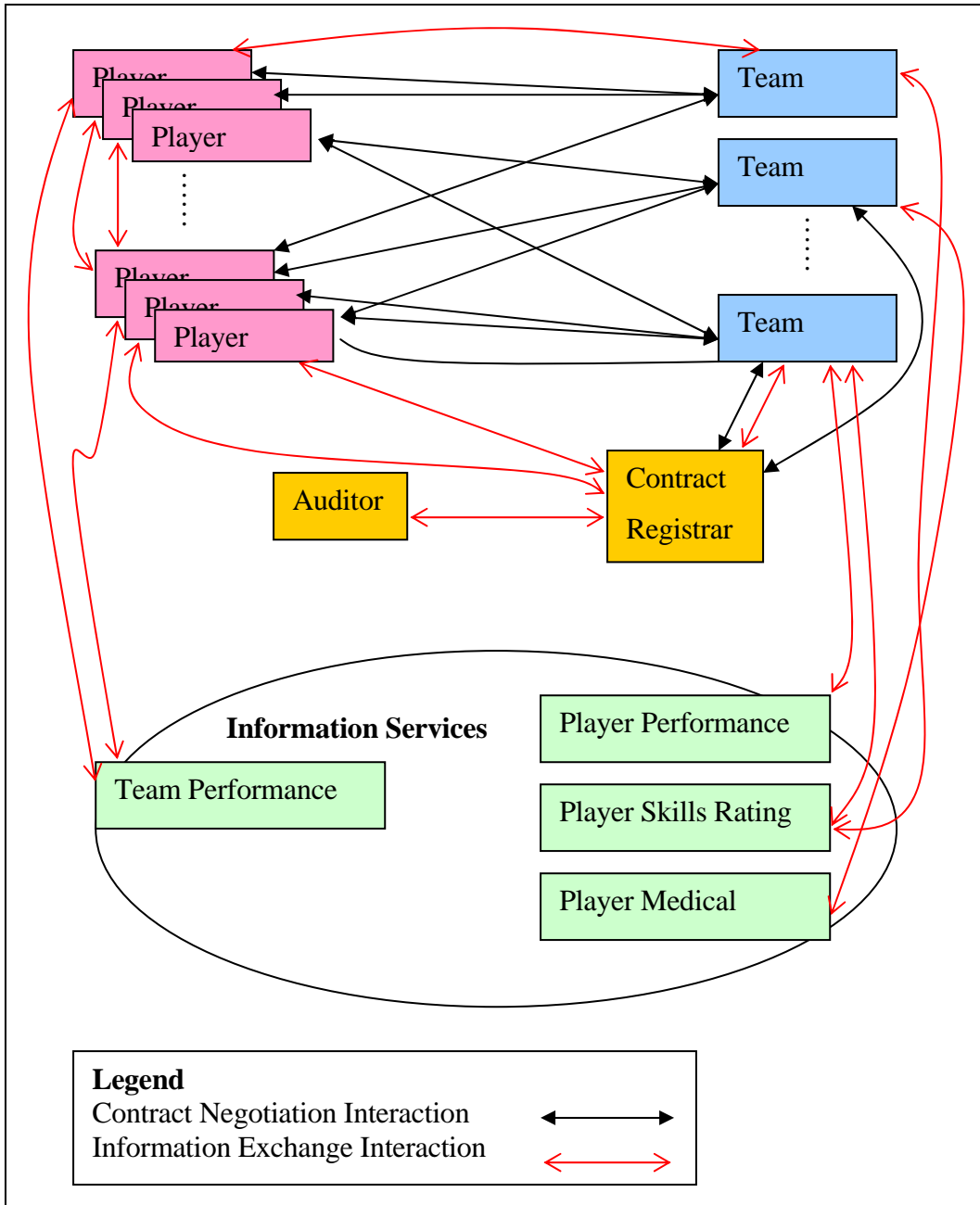


Figure 3

Interactions between agents occur solely through the exchange of messages. The transport through which the messages are delivered, as well as the messages

themselves, will be described in more detail in Section VII. The system defines two types of interactions between agents:

1. **Information exchange.** Information exchange may occur between any pair of agents. That is, any agent may request information (or any type) from any other agents; information requests may be declined, however. An important aspect of the information exchange mechanism in the contract negotiation system is that team owner agents and player agents (the two agent types that represent the interests of human users) may provide information to each other if their decision making algorithms indicate that a lie or bluff will prove beneficial; this allows the agents to conduct the negotiation process in a more “human” manner. However, no agents are permitted to provide false information to the services, whether they be informational or managerial. An information exchange interaction consists of an agent sending a data query message to another, and an optional response of a data message from that other agent. The types of data that may be queried, and where they may be obtained, are as follows:

- 1) **Player performance history:** statistics such as number of goals scored per game, number and type of penalties incurred, etc. These will be available from an information service.
- 2) **Player medical history:** injuries a player has sustained in the past, and other medical conditions that may impact a player’s game performance. Available from an information service.
- 3) **Player skills rating:** player skills will be divided into several categories and rated individually on some scale. The categorization

will be dependant on the sport in question, and may include passing skills, shooting skills, etc. Available from an information service.

- 4) **Player personal preferences:** essentially, information concerning what a player desires – what sort of team he wishes to play with, what sort of salary he requires, etc. These data originate from individual player agents, but depending on who they choose to share these information with, may be available from other agents as well. These data are *not* guaranteed to be correct.
- 5) **Team performance history:** statistics such as the number of games won in previous seasons. Available from an information service.
- 6) **Contract history:** details of a contract's life cycle, from it's initial offer by a team owner agent to a player agent to it's eventual acceptance, rejection, or withdrawal. Contracts that were officially accepted at some point in time are public knowledge, and data concerning them may be obtained from the Contract Registrar; contracts that were never accepted are considered private knowledge, and may only be obtained from other team or player agents.
- 7) **Agents/services list:** the Directory service will provide the addresses of all agents and services in the system.

2. **Contract negotiation.** This includes the offering of contracts and their acceptance or refusal, and occurs mostly between player agents and team owner agents. The contract registrar service would be involved whenever a contract is accepted or broken. The number of contract offers that the negotiation system and it's population of agents must deal with is large –

expected to be on the order of thousands to tens of thousands. So that these contracts can be easily referred to by the agents, each contract offer will be tagged with a unique ID. Team owner agents must request a unique ID from the Contract Registrar before making a contract offer. Contract negotiation interaction messages include the following:

- 1) **Contract UID request:** sent from team owner agents to the Contract Registrar to request a UID for a contract offer.
- 2) **Grant contract UID:** the Contract Registrar's response to a UID request.
- 3) **Contract offer:** sent from team owner agents to player agents; contains salary, cancellation penalty, and other details of a contract offer.
- 4) **Contract acceptance/rejection:** sent from player agents to team owner agents to accept or reject contract offers.
- 5) **Confirm contract acceptance:** when a team owner agent receives a contract acceptance message, it will send this confirmation message to the Contract Registrar to officially mark the contract as accepted.
- 6) **Contract withdrawal notice:** sent by team owner agents to player agents to notify them that a previous contract offer has been withdrawn. (and thus can no longer be accepted)
- 7) **Contract cancellation request:** an agent wishing to cancel an already accepted contract must send this message to the Contract Registrar. The cancellation penalty specified in the contract must be delivered to the other party involved before the contract can be

cancelled; however, the penalty delivery mechanism is not part of the contract negotiation system.

- 8) **Penalty receipt query:** acting on a contract cancellation request from an agent, the Contract Registrar will query the other agent involved in the contract as to whether the specified cancellation penalty has been satisfied.
- 9) **Confirm penalty receipt:** an agent's response to a penalty receipt query.
- 10) **Grant contract cancellation:** the Contract Registrar, upon receiving a penalty receipt confirmation, will mark the contract as officially voided and notify both parties to the effect through this message.

It is assumed that all agents in the system will be *persistent* – that is, their internal state will not be reset from year to year. Thus it is possible for the agents to refine their behavior based on experience/data collected in previous years. Also, there may be multiple implementations of the player and team owner agent software – in that case users of the system will have a selection of agent software (that makes use of different negotiation strategy, say) to choose from. The following sections describe each of the agent types in detail.

V.5 Player Agents

The player agent type is unique amongst the agent types in this contract negotiation system in that it is the only agent type that does not have a fixed, pre-defined set of objectives. This is due to the fact that a player agent represents the interests of a

player, who, being human, may have multiple, possibly conflicting objectives that can not be anticipated by software.

Each instance of the player agent type will act on the behalf of a single player. Fig. 4 is a player agent's potential internal structure. ("potential" since there can be more than one implementation of the player agent software.)

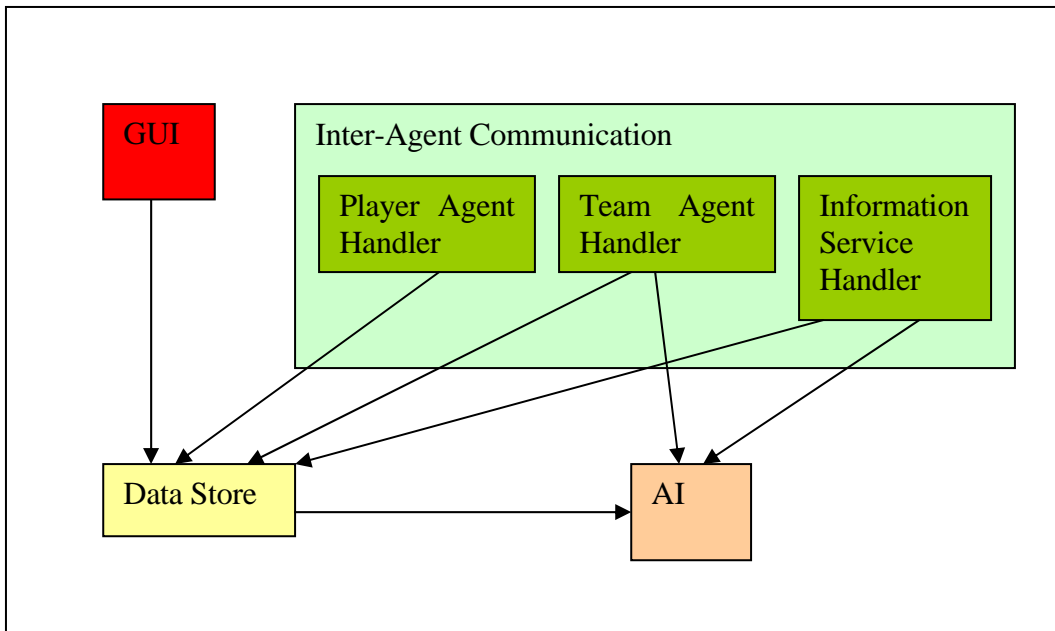


Figure 4 potential Player agent structure

The player agent implements a **GUI** module through which a user inputs his personal preferences. (and thus define the objectives of the player agent) These preferences include:

- **Minimum acceptable salary:** we anticipate that no players will wish to specify a *maximum* acceptable salary.

- **Home city preferences:** the user will rank the home cities of all the teams on a scale of 0 to 10; all else being equal, the agent will give priority to contract offers from teams whose home cities are highly ranked.
- **Individual teammate preferences:** the user, a player, will *rank each of the other players* on a scale of 0 to 10. During the negotiation period the agent will periodically query the Contract Registrar to determine which players each of the teams have already signed up; contract offers from teams that have already signed up players who are highly ranked as teammates will be considered more favorably. At the start of a negotiation period (when players have not yet begun to accept contracts) historical information may be used to determine which teams are likely to sign up which players.
- **Teammate performance preferences:** indicates whether the user prefers to be part of a team whose players are skilled (so that the team would have a better chance of making the playoff), or one whose players are (relatively) unskilled (so that the user will compare favorably to his teammates).
- **Team performance preferences:** indicates whether the user prefers to be part of a team that has a winning track record (again, so that playoff chances are better) or losing track record. (perhaps so that the user will not have to play as many games, thus increasing the salary to work ratio?)
- **Privacy/information sharing preferences:** the user will specify the fine-grained access rights to the Data Store of the agent. (the data store includes all the personal preferences specified above) Each piece of information may be made completely private, completely public, or selectively available to certain other agents.

The GUI is also responsible for informing the user of the events that has transpired so far in the negotiation process. Important events such as the arrival of contract offers (or notice of their withdrawal) will always be displayed; minor events (such as queries for information) may optionally be displayed.

It is important to remember the obvious fact that the player agent type's targeted users are professional athletes who will *not* want to invest a lot of effort into learning to use a piece of software – they will only adopt the contract negotiation system if it allows them to better concentrate on their chosen sports, and not make their life more complicated. Thus, a player agent's GUI must be designed with intuitiveness and ease of use in mind.

Incoming messages are parsed by the **Inter-Agent Communication** module and forwarded to various message handler modules depending on the source of the messages:

- **Player Message Handler:** messages from other player agents can only be information exchange messages. Data messages are forwarded to the Data Store; query messages will be checked against the information privacy preferences to see if the querying agent is to be allowed access to the queried information.
- **Team Message Handler:** contract negotiation messages will, of course, be forwarded to the AI module. Depending on the implementation, information exchange messages may be either processed directly or forwarded to the AI module – the AI module may be programmed to intelligently determine what information a team owner agent should receive.

- **Service Message Handler:** messages from an information service can only be data messages, and will be forwarded to the Data Store. Messages from contract management services will be forwarded to the AI module.

The **AI** module is responsible for the handling of contract negotiation messages; this is the most important module of a player agent, as the manner in which contract negotiation messages are dealt with will determine the success or failure of the agent. There are only three actions that may be taken in response to a contract offer: acceptance, rejection, or delay the decision until later. The AI module will determine which action to take based on the contract history of all the teams and the player's preferences. For example, if the received contract offer indicates a salary that is below average for the offering team or below the player's minimum salary preference, the AI module will likely choose to reject that offer, unless the contract histories show that most teams have already obtained a full complement of players for the current season, and thus the AI module could not afford the risk of not obtain a contract for the current season at all.

V.6 Team Owner Agents

The team owner agent type (and, by extension, the team owners that makes use of this agent type) is assumed to have only one objective – the maximization of profit. The profit that a team generates for any given season is a function of the amount of revenue generated through ticket sells and the amount of money paid out to the team's players as salary. (For the sake of simplification we will not consider other sources of revenue available to a team, such as the sales of team merchandise; nor will we consider the operating costs associated with maintaining a team of players)

Ticket sells is a function of the number of games that a team plays, which in turn is a function of the number of games hat a team wins; thus, a team owner agent will attempt to maximize the chance of winning games while minimizing the total amount of money spent on salaries. As is with the case of the player agents type, there may be multiple implementations of team owner agents; one potential implementation is very similar to the player agent structure proposed in the previous section, as can be seen in Fig. 5 below:

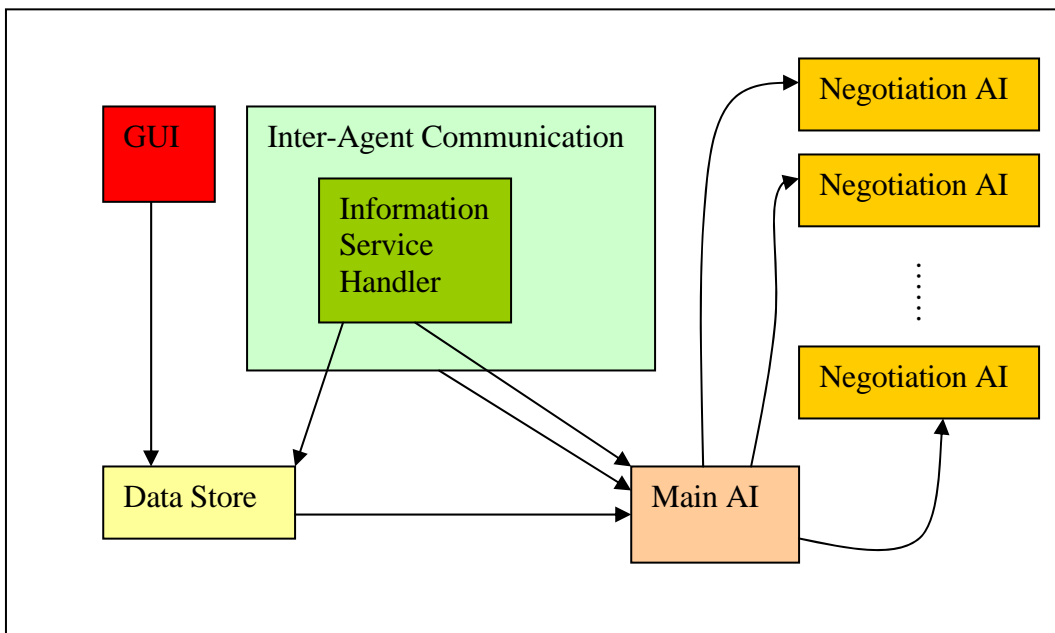


Figure 5 Potential Team Owner Agent Structure

This agent type's targeted users are team owners. It is assumed that team owners consider the management of their teams to be their main occupation; as such, they will be willing to invest considerably more time and effort to learn to use the contract negotiation system, so long as the results that the system generates betters those that they can obtain themselves. This is fortunate, as the team owner agent type's user interface will be much more complicated than that of the player agent

type – despite the simplicity of the team owner agent’s objective, it’s desired behavior cannot be configured merely through a list of preference selections.

As previously mentioned, a team owner agent will attempt to assemble a set of players that maximizes the teams’ chance of winning games. However, the agent does not possess knowledge as to what sort of player compositions will best serve this purpose. (Agents that can derive this knowledge through some sort of intelligent learning process is beyond the scope of this project) So, a team owner must provide his own knowledge (presumably learned through past experiences) of these matters to the agent through its **GUI** module.

The actual user input that is required will vary depending on the sport as well as the specific implementation of the team owner agent. (Some implementation may allow more fine-grained specification of desired team composition than others) Since this focus of this project is a contract negotiation framework for team-based sports in general, sport specific details of the GUI will not be described here. However, it will most likely involve the specification of the desired type and distribution of player skills amongst the team members. For example, the balance of shooting/scoring skills amongst a team’s players may be specified through a histogram adjustment interface (similar to those used by photo editing softwares to adjust exposures); this would allow a user to indicate a set of players all with similar, average capabilities in the skill, or a few players with extraordinary capabilities while the rest have below average capabilities, or anything between the two extremes.

A team owner agent’s **Inter-Agent Communication** module would likely not implement team/player agent message handlers (the way a player agent’s communication module does); all messages from other team and player agents will be forwarded directly to the Main AI module, which will intelligently decide how

information requests should be dealt with. The **Service Message Handler** is essentially the same as that of a player agent's.

In the potential implementation depicted in Fig. 5, The **Main AI** module determines which players are suitable team member candidates (according to a user's input) based on players' performance history, medical history, and skill ratings; these data can simply be obtained from the information services. Player specific **Negotiation AI** modules will then be spawned to initiate independent negotiation sessions with these players' respective player agents. The Negotiation AIs will base their negotiation strategy on the players' contract history (obtainable from the Contract Registrar) and personal preferences (only obtainable from other team or player agents, which are not *reliable* source of information). For example, a Negotiation AI dealing with a player agent that has a history of frequently canceling contracts will create contract offers with greater cancellation penalties to compensate; and if the Negotiation AI have knowledge of the highest salary that a player has been offered so far this season, it can create a contract offer that just tops it in order to obtain that player with minimum cost.

The **Main AI** module must also determine when it would appropriate to provide false information in response to a data query. For example, if data query for non-public contract histories is received from a competing team owner agent, the module may decide to respond with inflated salary figures to encourage said agent to create contract offers with too-high salaries and thus waste resource.

Different implementations of the team owner agent may employ different high-level strategies. An example of such a strategy may be to offer very favorable contracts to all the best players right at the start, thus denying these players to competing team owner agents and forcing them to fill their team with less capable

players, then canceling all these contracts at the last minute and so force these players to renegotiate for a lower salary.

V.7 Contract Registrar

There is only a single instance of the Contract Registrar agent. Its main role is to act as a facilitator for contract negotiation interactions; it has a secondary role of acting as an information service for histories of officially accepted contracts. It has a simple structure:

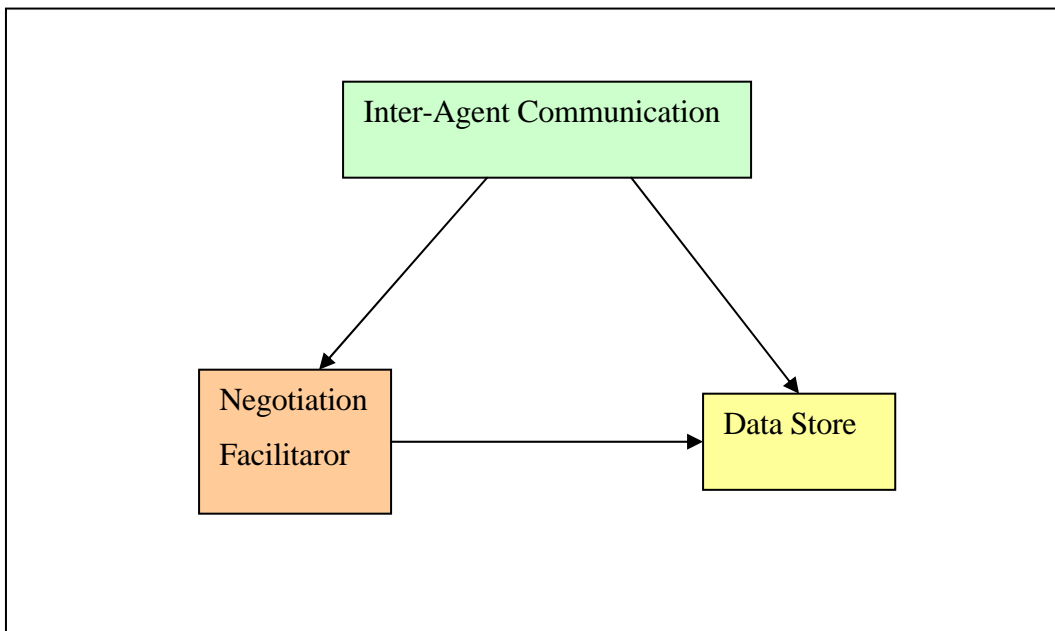


Figure 6 Contract Registrar structure

The **Inter-Agent Communication** module passes all information exchange messages to the **Data Store** module, and all contract negotiation messages to the **Negotiation Facilitator** module.

The Negotiation Facilitator will generate a UID in response to all contract UID requests – there is no reason to deny such requests. A contract acceptance confirmation message will simply be logged. When a contract cancellation request is received, the Negotiation Facilitator will query the other agent involved in the contract to see whether the cancellation penalty specified in the contract has been delivered. If the confirmation is received, a cancellation confirmation message will be sent to both of the agents involved.

Note that it is the responsibility of the two parties involved in a contract (as represented by a team owner agent and a player agent) to notify the Contract Registrar of contract acceptances or cancellations.

This agent does not interact with a user.

V.8 Auditor

This is also a single-instance agent. In theory, the Auditor agent's role is that of a law enforcement entity that ensures the four contract negotiation rules specified in Section IV are obeyed by all agents in the system. In practice, Rules Two to Four are essentially enforced by the Contract Registrar already; the Auditor agent's only task is to enforce Rule One, that is, the team budget cap. Thus, the Auditor has a very simple internal structure:

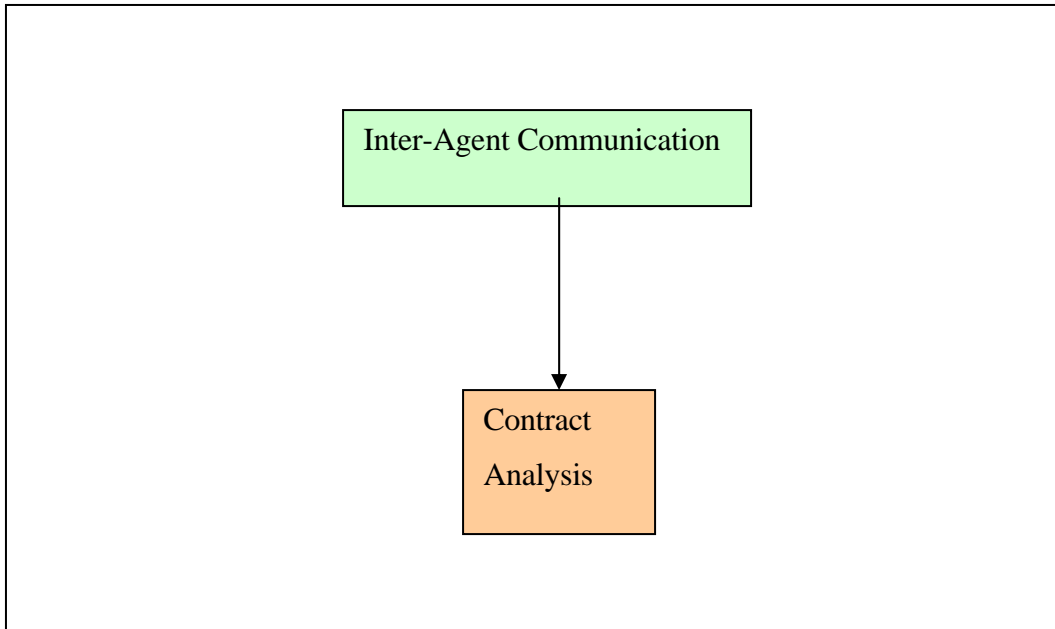


Figure 7 Auditor service structure

The **Inter-Agent Communication** module will periodically query the Contract Registrar for contract histories; responses to these queries will be forwarded to the **Contract Analysis** module, which will calculate the total expenditure of each team (the sum of all accepted contract's specified salaries *and* all the penalties that a team had to pay out for contract cancellations). Disciplinary measures for teams that exceed their budget cap are outside of the scope of the contract negotiation system.

Arguably the Auditor agent may be combined with the Contract Registrar, given its simplicity; however, implementing it as a separate agent would make it easier to add additional auditing capabilities in the future.

This agent does not interact with a user.

V.9 Directory Service

This single-instance agent provides allows agents in the contract negotiation system to locate each other. A directory service is often provided by existing multi-agent programming frameworks (such as FIFA-OS), so this project assumes that this service will be available “automagically”.

V.10 Information Services

Information services provide all agents with data that is considered “public knowledge”; these include player/team performance histories, player skill ratings and player medical history. These may be implemented as one large, monolithic information service/agent that contains all relevant data, or as a small number of separate agents each serving a different category of data. In either case, the agent(s) will consist internally of just a **Data Store** and an **Inter-Agent Communication** module.

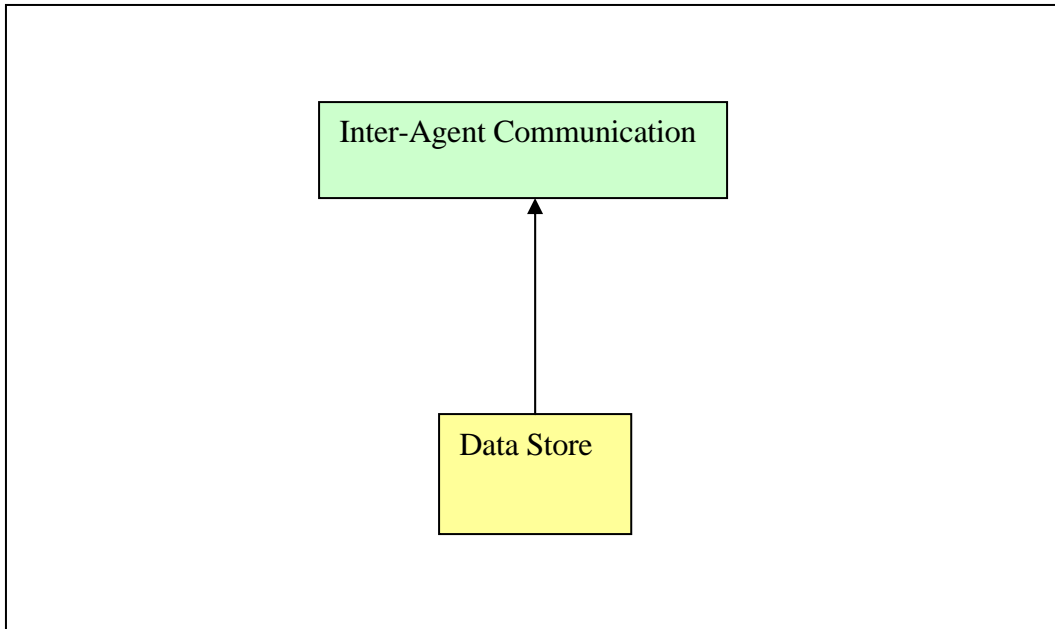


Figure 8 Information Service structure

VI Target Platforms

As the Use Cases in Section V indicates, the front-end of the contract negotiation system need to be able to execute on a large variety of platforms. A team owner agent's front-end must be available for Windows PC at the least, and Linux and MacOS support would be welcome. A player agent's front-end, in addition to these platforms, should also support mobile devices such as smart cell phones and wireless PDAs. On the other hand, service agents such as the Contract Registrar should ideally support popular server platforms such as Solaris.

One solution is to implement all agents as web services that use standard HTML pages as GUI front-ends. In this solution, the entire contract negotiation system

will exist on one or more web servers. This solution brings with it the added benefit that users of the system will not need to perform any software installation; it's disadvantage is that it places a much greater processing overhead on the servers.

The suggested solution is to implement the agents in Java; most of the target platforms mentioned above (including cell phones and PDAs) support some version of the Java Virtual Machine. This will also allow one to take advantage of existing Java based multi-agent frameworks, such as JAFMAS.

VII Inter-Agent Messaging and Ontology Considerations

Inter-agent communication must occur through the internet; the most sensible choice of message transport over the internet is HTTP over TCP/IP, as this allows one to avoid most problems with firewalls. The actual messages will take the form of XML fragments. A contract offer may look something like this:

```
<CNS_MSG>
<CONTRACT>
<CID>17238</CID>
<SALARY>$5,000,000</SALARY>
<PENALTY>$10,000</PENALTY>
</CONTRACT>
</CNS_MSG>
```

A data message containing a player's performance history may look something like this:

```
<CNS_MSG>
<PLAYER_PERF_HIST>
```

```
<NAME>Wayne Gretzky</NAME>
<PERF_RECORD>
<YEAR>1989</YEAR>
<TOTAL_SCORE>107</TOTAL_SCORE>
</PERF_RECORD>
<PERF_RECORD>
<YEAR>1990</YEAR>
<TOTAL_SCORE>153</TOTAL_SCORE>
</PERF_RECORD>
:
:
:
</PLAYER_PERF_HIST>
</CNS_MSG>
```

The transmission of these messages must be made secure; in addition, there needs to be an agent identity verification mechanism – we can't allow a team owner agent to pretend to be a *different* team owner agent and distribute contract offers that would lead to violation of the budget cap!

A unified ontology does not exist for any single sport, let alone team-based professional sports in general. The success of the proposed contract negotiation system depends in part on such a unified ontology, but constructing one will be exceedingly difficult. Some concepts in professional sports are easily defined and easily quantifiable – score counts and penalty counts are two such concepts. Most concepts, however, are either ambiguous or have multiple definitions depending on the sport. The task of goal keeping, for example, mean very different things and require very different skills in hockey and soccer; and the definition of skill ratings

can not easily be agreed upon. (Exactly what constitutes “good” or “excellent” defense skills?) The problem of ontology will likely prove to be the biggest obstacle to the realization of the contract negotiation system.

VIII Possible Future Expansions

If the initial implementation of the contract negotiation system proves successful, the system’s scope may be extended in these ways:

- Player agents may be granted the ability to make “counter offers” to team owner agents; this would allow player agents to be more pro-active in the negotiation process, as well as provide team owner agents with another indication of the personal preferences of the players.
- The team owner agents can incorporate additional intelligence to automatically determine what the most successful team composition would be; this could potentially eliminate the need for user input from the team owners completely. Mostly like the team owner agent GUI would still need to provide a means for the team owners to manually affect the negotiation process, if only for psychological reasons – users often prefer to have a semblance of control.
- Introduce a more complex model of profit into the team owner agents. This model will take into account additional profit sources such as the sales of team merchandise, as well as other factors effecting game ticket sells – the popularity of players, for instance. This may lead to interesting high-level strategies such as selecting players based on popularity alone – the resulting

team may not make the playoffs, but the increased ticket sells per game due to the players' popularity may offset the decreased number of games.

- Add a banking service to the contract negotiation system. Salaries payments and contract cancellation payments can then occur within the framework of the system, which will make the entire negotiation process more efficient. The Contract Registrar and Auditor services will now also have additional ways to verify that the player and team owner agents are obeying the four contract negotiation rules.

IX Discussion

If the ontological hurdles can be surmounted and the contract negotiation system actually implemented, observing the player and team owner agents' changing behavior over time should prove most interesting. As the agents gain experience and began to adept to other agent's actions, will all agents of a given type converge toward a set of similar strategies? Or will they continue to maintain (or even develop) uniqueness? In other words, in the artificial "ecosystem" that the contract negotiation system provides, will we see signs of convergent evolution, where initially unique organisms gradually gain similar characteristics, or will there exist many specialized niches in which diverse behavioral patterns may exist?

In fact, the agent-based system proposed in this document may have more value as a simulation system for the study of the evolution of human behaviors in social settings than as an actual contract negotiation system. This is especially true if one considers the fact that, by converting the system from one to be used in the real

world to one that simply needs to simulate an artificial environment, much of the ontological issues disappear – it is no longer necessary for the system’s ontology to be in agreement with the ontology of the complex and ambiguous world of professional sports.

X References

Wooldridge, M.; Jennings, N. R.; Kinny D. (2000). *The Gaia Methodology for Agent-Oriented Analysis and Design*. *Autonomous Agents and Multi-Agent Systems*, 3, 285-312, 2000.

Chauhan, Deepika (1997) *JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation*. ECECS Department Thesis, University of Cincinnati

<http://www.nhl.com>

<http://mlb.mlb.com>

<http://www.nfl.com>