UTUtd 3D Development Team Description Paper

Mehrdad Afshari and Sahar Asadi

School of Math, Statistics, and Computer Science,
University of Tehran, Iran
mehrdad@devct.com
asadi@khayam.ut.ac.ir
http://www.fos.ut.ac.ir/ robocup

Abstract. This is team description of UTUtd for RoboCup06, Suzhou. Looking back to Simulation Leagues in previous years makes the need for a reliable, adaptable and extensible league manager pretty clear. While analyzing the past RoboCup competitions and based on our experiences in Soccer Simulation League, we decided to implement a *league-independent*, *cross-platform League Manager* specifically optimized to hold 3D Soccer Simulation competitions. Unlike simple scripts, this League Manager is designed to eliminate league's organization problems such as handling logfiles and results all together automatically. This League Manager can even make it easy to have SSIL 3D competitions.

1 Introduction

UTUtd is the RoboCup team of School of Math, statistics, and Computer Science, University of Tehran. This team has started its activity in 2D since 2000. After that a new team based on previouse experiences started its research on 3D since the very begining of Soccer Simulation 3D in 2004. Continuing our research, UTUtd got interested in the development of the simulator and consequently got champion of 3D Development competitions in Robocup06, Bremen. Participating in many RoboCup events, we have found it essential to have a reliable and extensible League Manager. An adaptive League Manager can be used also in other leagues as a general platform; in this way, it even can be a step toward achieving general goals of RobCup's roadmap.

In the Soccer Simulation League, teams challenge each other by running their agents in a simulated soccer environment. The primary component of the simulation environment is the *SPARK Simulation Server* [**?**]. This component simulates a real-world soccer game. Agents, which are developed by participating teams, connect to the server to compete against each other. The server is responsible for providing the game state data to the agents and updating the game state according to the actions submitted by them. To visualize the soccer game, a *monitor* connects to the server and receives the game data from it. As noted earlier, the server handles the primary responsibility of what happens in the game and enforces soccer rules but it is not able to handle all the game

rules, therefore, in the competition, a *human referee* oversees the simulation. The server also saves complete game information in a *log file*.

In the actual league, team binaries are usually run on separate machines and communicate with the server over the network. In order to start a match, an instance of the server is launched and waits for agents to connect. After starting the server, the binaries located on a shared directory are run with and provided with the network address of the server they are going to connect to. A monitor will also connect to the server to visualize the match. After the match is finished, the board should be updated to reflect the result of the match. Since some teams may participate remotely, its crucial to update the board instantly for them to see at the same day. They also need the log files to analyze the match that their agents played. In previous years, somtimes the Organizing Committee had to handle and control all of these manually.

We need an infrastructure to manage leagues automatically, while ensuring the reliability of the environment. Since the *League Manager* runs the binaries provided by the teams, it cannot rely on their correctness (i.e. binaries can have problems like not starting correctly, not exiting correctly after the match). The infrastructure has to be flexible enough to handle situations such as a failed startup of a specific team.

Due to time restrictions in the league, matches are run in parallel on different clusters which makes the league structure more complex. League manager has to be able to deal with arbitrary complex league structures.

An important requirement in this project is to be able to update score board as soon as a match ends and make it publicly accessible on the Web. Without an automated process, this will require manual work to update the board and to keep and maintain log files. Therefore, we are providing the capability to manage log files right into the League Manager.

In addition to these goals, we want this system to be as league-independent and platform-independent as possible. We also wanted to keep the architecture simple and extensible enough to allow further changes as the need arises. This will make it easier to add features, such as creating and providing videos of matches and making them available on the Web.

Section **??** describes architecture of League Manager System and its components in details. In Sec. **??** techinical points about the implentation are presented. We are working on this League Manager and we paln to prepare it as a usable utility for 3D Competitions in Suzhou. In Sec. **??** future works are explained.

2 System Architecture

The League Manager consists of several independent modules that communicate over the network:

- League Coordinator
- League Drivers
- League Manager User Interface
- A set of plugins

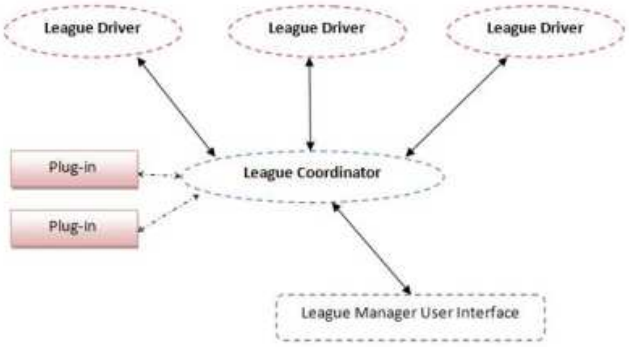

Fig. 1. System architecture of the Leage Manager

2.1 League Coordinator

As it is shown in Fig. 1, League Coordinator is the main component of the system and runs as a service on a single computer. This component manages the league structure. It sends commands to other components in order to run the league and receives reports of the matches played.

#	Team1	Team2	Status	Start	Stop
1	Team 1	Team 3	Pending	Start	Stop
2	Team 2	Team 4	Pending	Start	Stop
3	Team 1	Team 5	Pending	Start	Stop
4	Team 2	Team 6	Pending	Start	Stop
5	Team 1	Team 7	Pending	Start	Stop
6	Team 2	Team 8	Pending	Start	Stop
7	Team 3	Team 5	Pending	Start	Stop
8	Team 4	Team 6	Pending	Start	Stop
9	Team 3	Team 7	Pending	Start	Stop
10	Team 4	Team 8	Pending	Start	Stop
11	Team 5	Team 7	Pending	Start	Stop
12	Team 6	Team 8	Pending	Start	Stop
13	Rank 1 of group "Group A" in round "Round 1"	Rank 2 of group "Group B" in round "Round 1"	Pending	Start	Stop
14	Rank 2 of group "Group A" in round "Round 1"	Rank 1 of group "Group B" in round "Round 1"	Pending	Start	Stop
15	Rank 1 of group "Group A" in round "Round 2"	Rank 1 of group "Group B" in round "Round 2"	Pending	Start	Stop
16	Rank 2 of group "Group A" in round "Round 2"	Rank 2 of group "Group B" in round "Round 2"	Pending	Start	Stop

Fig. 2. A sample for defining team in the League Manager

League Manager has the responsibility of updating the score board and managing log files. It also keeps track of the league schedule and runs matches automatically based on the schedule. In Fig. 2, a sample of managing games in league and status of games can be seen.

To achieve the extensibility goal, league coordinator is implemented in an event-driven, plugin based fashion. It raises events which plugins can handle and extend the system by doing specific tasks.

2.2 League Drivers

League Drivers run on machines that are going to run the simulation server and agents as services applications. They are responsible for starting and stopping the server and the agents. They should also report back the result to the League Coordinator.

2.3 League Manager Client

In order to define the league structure, we control the League Coordinator service by means of a user interface program which is implemented as *web-based* application.

Fig. 3. Features of League Manager to handle teams and games

This application will let us define teams, the user account they are going to use, the home directory containing their binaries as presented at Fig. 3 (a). Moreover, it helps us define league structure, clusters as shown in Fig. 3 (c), that are going to run the matches, match groups, as shown Fig. 3 (b) and basically any functionality required to define a soccer league structure.

It will also give us the ability to start and stop matches.

2.4 Plugins

League coordinator is implemented with extensibility in mind. It has the ability to dynamically load plugins specified in its XML configuration file.

When plugins are loaded by the core module, their initialization procedure is called. In this procedure, modules can subscribe to the events provided by the core system and perform a custom procedure on each event. For example a Web upload plugin registers itself to execute a procedure when a match is completed and uploads its log file to a server accessible over the Web.

3 Technical Infrastructure

As one of our project goals, we wanted this system to be available on major platforms such as Linux, Mac OS X and Windows. To achieve this goal, we selected the free-software implementation of CLI runtime, Mono project, which is licensed under LGPL as our platform.

4 Future Works

Considering the past RoboCup competitions, it is quite obvious that simple scripts that are made to manage leagues do not have the capability to be used in real-world scenarios. Instead of reinventing the wheel, we decided to go for a complete and adaptable platform that can be actually used. The extensibility of this platform made us to think about extra plug-ins that might be useful. We mention a couple of them here.

One of our ideas is to implement a plug-in for this system to be able to broadcast matches live over the Web using Adobe Flash or Microsoft Silverlight platforms.

Another interesting idea is being able to integrate competition news with the match data and publish it on the Web site together with the score board and game schedule. Its even possible to publish all these data on the Web site using SportsML [**?**] format with a simple plugin.

UTUtd2006 released a package for monitor and coach for the previous simulator -the one with sphered agents-[**?**], but they are general and extensible to be used for simspark simulator, too. The idea is to use package for creating a monitor with extra features which can be connected to League Manager. The monitor will send statistics of games to the League Manager. These processed data can be either released as a report from games or an analyzer can use it for team learning.

References

1. Simspark RoboCup Soccer Simulator available at http://simspark.sourceforge.net/wiki/index.php
2. Sports Markup Language documentation available at http://www.sportsml.org/documentation.php
3. Aghaeepour, N., Bastani, M., Disfani, F.M., Masoudnia, S., Radpour, S., Siahpirani, A.F., UTUtd2006-3D-dev Team Description Paper (2006), http://www.fos.ut.ac.ir/ robocup/download.htm