THE PROMUS AGENT SYSTEM, A TOOL TO ASSESS A VIRTUAL TEAM’S PERFORMANCE

Mauro Nunes
ADETTI/ISCTE
Av. Forças Armadas 1649-026 - Lisboa, Portugal

Henrique O’Neill
ADETTI/ISCTE
Av. Forças Armadas 1649-026 - Lisboa, Portugal

ABSTRACT
This paper introduces a study about the application of intelligent agents on managing the performance of virtual teams. The motivation behind this study comes from the need for better measuring and management tools of team performance. On previous work, an agent mediated team performance model was defined and is now tested with an agent system. We present the Promus agent system, which was used as a test bed to evaluate a virtual team’s performance on trial scenario based on a distribution game (Beer Game).

KEYWORDS
Virtual Team Teamwork Performance Agents

1. INTRODUCTION
Virtual Teams are strongly dependent on technology (information and communication) to achieve a good level of teamwork (Townsend et al., 1998). So far, technology has been seen only as Computer Mediated Communication (CMC), with the passive role of being just a communication tool for team members. Although, some progress has been made since the time when, for instance, only Videoconference, Shared Application or Whiteboard tools were available (see Hofte (1998 p. 9) for more details). Nowadays, these tools are being integrated in rich and complex online workplace environments, to aid team members collaborate (Schmidt and Rodden, 1996; Carstensen and Schmidt, 1999). Nonetheless, these new workplaces are not targeted towards team performance and do not proactively manage a team’s performance. It requires a technology that autonomously and proactively evaluates a team’s performance, being also able to provide feedback to the team. Autonomy and proactivity are fundamental behaviours of the Agent Technology (Knapik and Johnson, 1998 p. 3), a branch of the Artificial Intelligence research.

Agent Technology provides new opportunities and challenges to be explored in teams. Researchers have applied this technology at different levels in a team, like on teamwork simulation (Tambe, 1997), team communication (Yen et al., 2001) and team selection/formation (Tidhar et al., 1996). Based on a comprehensive review of research about human performance on team settings, Paris et al. (2000) maintains that there is still a need to “develop more dynamic measurement systems that allow for on-line assessment of teamwork”, particularly because the observational methods (i.e. behavioural checklists or videotaped observation, etc.) have been insufficient to measuring teamwork processes (Chung et al., 1999). These methods are labour intensive and time consuming, which decreases the speed of analysis and reporting of team performance, becoming unsuitable for large-scale test settings and to fully capture the dynamic nature of teamwork. We believe that agents may offer a value added contribution to such a measurement system and also extend it, by proactively managing the performance.
It should be noted, that our approach is focused on the synergistic relationship between intelligent agents and teams composed by humans. This relationship is established with the purpose of ensuring the team’s performance. We have formalised this relationship with the aTeam model.

1.1 aTeam – Agent Mediated Team Performance Model

The aTeam model extends the normal team environment by integrating a set of team responsive intelligent agents, thus creating an agent mediated work environment. Our assumption is that these agents are able not only to observe all team processes inside the environment, but also to cooperate with it by assisting the team’s performance. Performance evaluation is done through a time constrained recurring phase model of team processes. The aTeam model has been formally defined on previous work (Nunes and O’Neill, 2004) and is the base for this study.

To test the aTeam model it is required a valid testing tool. According to Paris et al. (2000), “team assessment tools should: (1) identify processes linked to key team outcomes, (2) distinguish between individual and team level deficiencies, (3) describe interactions among team members to capture the moment-to-moment changes that occur, (4) produce assessments that can be used to deliver specific performance feedback, (5) produce evaluations that are reliable and defensible and (6) support operational use.”. Having in mind these requirements, the Promus agent system has been designed and implemented.

1.2 Related Work

From the literature it is possible to identify scientific work related with this topic. The work from Lenox et al. (1999) has some resemblances with the proposed, by using agents to support team performance. However, on their study no geographically dispersed teams are considered, which decreases the communication and coordination effort. The agents also work as specialized facilitators rather than representing each member working as a team, circumscribing the work of the intelligent agents to support and promote teamwork dimensions (Team situation awareness, Support behaviour, Communication and Team initiative/leadership) in a target identification task.

The approach from Chung et al. (1999) towards the development of a real-time measurement system for teamwork skills is also related to our approach. They focus their work on assessing team processes used by a group of individuals responsible for jointly constructing a knowledge map. Team processes were measured according to a taxonomy developed from previous work (see O’Neil et al., 1997), likewise, we too measure team processes according to a similar taxonomy (see Marks et al., 2001). However, their system was not proactive towards team performance, rather just an automatic measurement system. Although a team member identity was anonymous, they were all collocated at the same time, whereas in our study we are imposing time dispersion to team members, as it is a key characteristic in a Virtual Team.

The work from Miller and Volz (2001) also relates to our study, in the sense that they use a model of teamwork to identify weaknesses in team interaction skills in a human trainee within a team of agents. Although their study’s subject is focused on training mixed teams of agents and humans, they state that it could also be used as a tracking or monitoring tool for human teams. They have developed and implemented a multi-agent architecture called CAST (Collaborative Agents for Simulating Teamwork), which simulates and supports teamwork between agents.

The remaining of this paper describes the developed agent system and its application to assess a virtual team’s performance. In particular, Section 2 describes the Promus agent system and a proof of concept. Section 3 demonstrates how the agent platform was used for measuring and managing team performance. Finally, Section 4 evaluates the platform based on user trials.

2. THE PROMUS AGENT SYSTEM

The Promus Server (PS) was developed in JAVA and was loosely based on the agent framework described by Kendall et al. (2000). The server is a multithreaded agent environment with task and state oriented agents and uses an XML messaging protocol to communicate with the outside world. Currently, the server is running with two types of agents: Personal Agents (PA) and Services. A PA is an agent that represents a
human team member based on a profile. A Service is a specialized agent that supplies a specific service, for instance the access to a calendar server is provided by a Calendar Service, while a Profile Service provides access to a profile database. New services can be easily attached to the server:

The agent architecture was divided into three layers: Sensing Layer, Reasoning Layer and Collaboration Layer (Figure 1). The Sensing Layer is responsible to sense the PA environment and feed the Reasoning Layer with this sensory information. The Reasoning Layer has the purpose of concentrating all the business logic and rules for the PA to carry out a defined set of tasks. This task orientation provides the means to encapsulate the knowledge of a specific domain (i.e. teamwork performance assessment, awareness, calendaring, etc.). The Collaboration Layer is responsible for managing the collaboration between the PA and its environment.

Figure 2 shows the conceptual architecture of the Promus Server.

System design followed an object-oriented approach based on software design patterns (i.e. mediator, observer or factory pattern, more details in (Gamma et al., 1995)). It was designed having in mind requirements like scalability, extensibility and cohesion. This has lead to an approach where the core agent system relies on the services from abstract classes. Figure 3 shows some of the system classes, using a UML1 class diagram. Currently, the Promus Server is composed by 85 classes, which are all bound to the PromusComponentEnvironment root class. All components managed through the PromusComponent class.
To prove that the developed agent system was able to cope with some degree of complexity and flexible enough to allow for different applications, we have developed an enhanced version of an instant messaging application, the Promus Client.

2.1 Promus Client – A Proof Of Concept

The Promus Client was developed in order to test the possibility of the Promus Server to sustain complex and diverse applications. This application provided the user the following services:

- Buddy awareness – The PA manages a list of users and notifies the user when anyone on the list is on-line.
- Chat – Initiate chat sessions with several users.
- Offline messaging – Send a message that is delivered by the destinee’s PA when it gets online.
- Calendar – The PA is able to manage the user’s calendar and is aware of the appointments.

These services were supported by the platform by extending the PA’s knowledge with specific managers for buddy management, offline messaging and calendar management. It were also created the services of Chat, which provided the chat functions in the platform, and Calendar that created an interface to a remote Calendar Server (Netscape).

It was developed using Visual Basic (Fig. 4 shows the interface) and HTML generated from XSP\textsuperscript{ii} under a Tomcat/Cocoon server, which could also connect to the Promus Server. This architecture is depicted in Figure 5.

This application was used at the development stage and also on a daily bases with users outside the development team. It helped us to conclude that:

- It is possible to keep connections for extended periods of time and detect when a connection was dropped, reflecting this change into the platform components.
- The platform was self-maintained with low levels of memory garbage and resilient to errors.
- The messaging protocol was acceptable since it is based on serializing complex message objects into XML, which can generate large messages and create communication problems on reduced bandwidth.
- Sort out concurrency issues on the multithreaded code.
- Server performance was acceptable, although it decreased when a high number of users were active at the same time. Performance analysis indicated that this bottleneck could be improved with a different hardware setup and some code tuning.

After this proof of concept, the Promus Server was ready for introducing team performance management functions and to be the study test bed.
3 AGENTS AND TEAM PERFORMANCE

To suitably evaluate the proposed team performance model mediated by agents it must be applied in a test case which, under controlled laboratory conditions, can help capture the behaviour of virtual teams. This test case must allow team members to be geographically and/or time dispersed, provide suitable performance data and be fun to play. We have selected the Beer Game as a test case, inspired by the approach from Rafaeli and Ravid (2003), which used the game with virtual teams and from Kimbrough et al. (2002), which had intelligent agents playing the game. It was developed by the Systems Dynamics Group of the Sloan School at MIT and has been described by Sterman (1989; 1992). The game simulates a distribution system where the team’s objective is maximizing net profit. Each team member takes a position (Retailer, Wholesaler, Distributor or Factory) in the supply chain

3.1 The Beer Game test case

The beer game was originally created as a board game. Recently, computerised versions have been created, enriching this system dynamics experience. Our version follows the original game specification, but with some enhancements to make it suitable for the study of virtual teams. In particular, the following enhancements have been made:

- Day duration – A simulated day lasts at most 2 minutes. If no play is made, then a standard play is assumed and the game moves to the next day.
- Player offline – On each day a player is randomly set as offline. It can only communicate through mail, which is only delivered on the next day. This is used to emphasize time dispersion, a common characteristic of a virtual team.
- Chat meetings – Online players can create chat meetings to discuss game issues and strategy.
- Demand with a seasonal pattern – A more complex demand pattern was used in order to make the game harder and foster team cooperation.

The game engine was implemented as a service in the Promus Server and the client was developed in Visual Basic (Fig. 6 shows the interface). The player’s agents were also extended with game knowledge, enabling the agent to play the game if a user was not online (it also enabled to run games only with agents).
The picture above (Fig. 7) also shows the Performance Service, which was created to manage game performance data. It also interfaces with the performance database that is outside the server. Before using the platform on real trials, it was subject to a workload performance evaluation. This consisted on running worst-case scenarios, particularly when having multiple players concurrently playing the game. As an example, the platform was subjected to a testing scenario where two teams, composed only by agents, played the game. Unlike humans, agents play the game almost instantly, which increases the requirements for load balancing and synchronization. The next graph (Fig. 8) shows a workload evaluation based on the available thread-pool workers.

Figure 8. Workload evaluation – Available thread-pool workers

From the graph it is possible to see that, on a heavy workload (games running), the number of available workers is never exhausted and the system provides a reasonable response to the extra processing demand. Available thread-pool workers is one of the most relevant measures of performance, since it is the driving force of the Promus agent platform. If no workers were available the platform would not be able to accept new processing requests.

4 EVALUATION

The Promus Beer Game implementation was used on four trials. The first three trials consisted on a challenge for undergraduate (Management) students, where, on each, 20 participants in 5 teams played the game at the same time (60 participants in total). These trials were run on the same network of the Promus Server. The last trial consisted on a challenge for postgraduate (Logistics) students and had 12 participants in 3 teams. This trial was run on a remote location outside the Promus Server network.

Each trial took on average 1h10m (36 game days) and although the players were at the same room they were not allowed to engage on oral communication. The Promus Server successfully supported the game trials, processing more than 22500 system messages, generating 12825 performance data records and 2192 game plays.

The Promus server enables to post process game data by simulating the game behaviour and sending the performance data to the player’s agents. The agents use the performance data to calculate the individual performance index. Each performance measure value is also compared with the expected performance. Figure 9 shows the performance manager interface.

Figure 9. Promus Performance Manager interface
The expected performance is calculated using a dynamic reference value based on the historic data stored on the performance database. The individual performance data is normalized based on the expected value and the deviation is evaluated using fuzzy logic. If the deviation is considered high enough (fuzzy function), then a performance warning is raised. The next sequence diagram depicts this user performance evaluation process (Figure 10). The method `evaluatePerformanceData()` contains the fuzzy evaluation of the performance data. The resulting evaluation is then sent to the Performance Service, which will store the data and, if logged in, show it in the Promus Performance Manager interface.

Trial evaluation shows that the agents are able to critically assess both individual and team performance, particularly in the form of performance warnings.

Unlike normal performance evaluation tools, the Promus agent platform conducts performance evaluation in real time, providing performance warnings while the task is being carried out. This potentially enables a higher degree of control to a team manager (a target client for this type of applications), which can perceive how a team is currently performing and how probably they will perform in the future, hence acting almost as an early warning system. Practical applications of this research can be, for instance, evaluating the performance of operators in a Call Center, where agents monitor the performance of each team of operators and provide real-time feedback to team managers, not only with the potentially under performing operators, but also with the work patterns of the better performing.

This provides experimental validity to the proposed agent mediated team performance model and its basic assumption, that by introducing agents into a team it is possible to critically assess individual and team performance, which may help improving the performance of potentially under performing teams. However, there are some limitations, namely, the test case does not take into account all the variables that are present on real life virtual team settings, the agent’s performance evaluation can be improved to include agent cooperation, social and ethical implications on having agents monitoring a person’s performance. These issues will be pursued on future work.

5 CONCLUSION

We have presented the Promus Agent system as an application of intelligent agents on managing the performance of virtual teams. This system was used as a testing tool to validate the aTeam model, which has been proposed on previous work. It enabled to run user trials on a test case scenario in which virtual teams played a computerised version of a distribution game (Beer Game). The agents were able to assess both individual and team performance. Hence, the agent platform fulfils the initial requirements from Paris et al. (2000) for a team assessment tool. In particular, because it can (1) assess both individual and team performance, (2) perceive and capture team processes, (3) provide performance feedback in terms of performance warnings, (4) justify performance assessment by showing the fuzzy rules evaluation and (5) can be applied under real circumstances.

ACKNOWLEDGEMENT

This research work has been partially supported by Associação para o Desenvolvimento das Telecomunicações e Técnicas da Informática (ADETTI) and Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), under the scope of the UNITE project (IST-2000-25436).
REFERENCES


i www.uml.org

ii XSP stands for eXtensible Server Pages and is one of the core technologies available in Cocoon. Cocoon is one of the seven current parts of the Apache XML project. The new Cocoon paradigm is based on the fact that document content, style, and logic are often created by different individuals or working groups. Cocoon aims for a complete separation of the three layers. In other words, Cocoon allows content, logic, and style to be separated into different XML files, and it uses XSL transformation capabilities to merge them.