Analyzing Multi-agent Activity Logs Using Process Mining Techniques

A. Rozinat[1,2], S. Zickler[2], M. Veloso[2], W.M.P. van der Aalst[1], and C. McMillen[2]

Abstract

Distributed autonomous robotic systems exhibit complex behavior that—although programmed, but due to the impact of the environment—only materializes as the process unfolds. Thus, the actual behavior of such a system cannot be known in advance but must be observed to be evaluated or verified. In this paper we propose to use *process mining* techniques to extract, compare, and enhance models of the actual behavior of a multi-agent robotic system through analyzing collected log data. We use the example of robot soccer as such a multi-agent robotic system, and we demonstrate which types of analysis are currently possible in the context of the process mining tool set ProM.

1 Introduction

Robotic systems are growing more and more complex as they seek, for example, to be self-reconfiguring, self-organizing, or working in teams. While their behavioral logic is of course programmed, and, thus, in principle predictable, the more autonomous a robot grows, or the more it adapts to its environment, the more it is true that the actual behavior of the system cannot really be known in advance anymore. For example, in a team of robots, the overall system behavior is determined through interaction among multiple robots. Or, if robots interact with humans, their actions are influenced by the actions of the human. Thus, the question whether this overall behavior corresponds to the intended system behavior can only be answered through observation.

Process mining techniques use log data to analyze *observed processes* and have been successfully applied to real-life logs from, e.g., hospitals, banks, municipalities etc. (see [2] for one of many real-life applications). The basic idea of process mining is to discover, monitor and improve real processes (i.e.,

Information Systems Group, Eindhoven University of Technology, NL-5600 MB, Eindhoven, The Netherlands. {a.rozinat,w.m.p.v.d.aalst}@tue.nl · Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213-3890, USA. {szickler,veloso,mcmillen}@cs.cmu.edu

not assumed processes) by extracting knowledge from event logs. Today many of the activities occurring in processes are either supported or monitored by information systems. However, process mining is not limited to information systems and can also be used to monitor other operational processes or systems, such as complex X-ray machines, high-end copiers, or web services. The common denominator in the various applications of process mining is that there is a notion of a process and that the occurrences of activities are recorded in so-called event logs [1].

In this paper, we use log data collected by the CMDragons team during the international robot soccer competition 'RoboCup' 2007 to investigate the applicability of process mining techniques to a multi-agent robotic system. In the context of robot soccer, the motivation for analyzing log data is two-fold:

- **Self-analysis**: While detailed logs are recorded during the competitions, the evaluation of a game is carried out mostly through watching the accompanying video recordings. A wealth of detailed data are available, but usually they are not analyzed in a structured way. We argue that a systematic and more high-level analysis of these log data can help to obtain an overall picture, and to, for example, discover transient faulty states that—due to being held only for a short time—are difficult to be observed by the human eye.
- **Opponent analysis**: Obtaining behavioral models of the opponent robots is even more interesting as their intentions are hidden and the only way to derive knowledge about their strategic behavior is through observation. We envision the application of process mining techniques in combination with activity recognition [9, 7], which is able to identify high-level actions (for example that a robot is "attacking") based on low-level behaviors (such as the robot's position and velocity).

Without loss of generality, we can restrict ourselves to self-analysis, since the described techniques can be equally applied for the purpose of opponent analysis given that appropriate activity recognition mechanisms are in place.

The paper is organized as follows. First, we describe the domain of robot soccer that is used in the remainder of this paper as an example of a multi-robot system (Section 2). Then, we introduce process mining and explain how process mining can be applied in the context of this example domain (Section 3). Next, the log data that are used as input for the process mining techniques are described (Section 4) and some illustrative analysis results are presented (Section 5). Section 6 concludes the paper.

2 Robot Soccer: A Multi-agent System

Behavioral multi-robot systems are control architectures where multiple agents coordinate the execution of different individual tactical approaches,

called behaviors, typically in order to achieve a common goal. One particularly interesting behavioral multi-agent domain, which we are going to use as the data source for our experiments, is the RoboCup Small Size League. Here, the objective is to have two teams of small robots compete in a game of miniature soccer, without human intervention. In the Small-Size league, each team normally consists of five homogeneous, omni-directional robots which are remotely controlled by an off-board computer. The computer obtains its observations from two overhead cameras mounted above the soccer field which are then processed to provide very accurate estimates of positions and orientations of all the players on the field. Challenges of this domain include its very fast pace, and its complex tactical properties. A scene from a typical robocup game is shown in Figure 1.

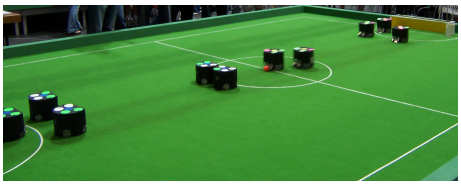

Fig. 1 A scene of a RoboCup Small-Size League game.

This paper utilizes real game data collected by Carnegie Mellon University's Small-Size team "CMDragons" [5]. The software architecture of the team's offboard control system is shown in Figure 2. The server component in this diagram performs computer vision and manages communication with the robots. The intelligence of the system arises from the soccer component which embodies all autonomous decision making processes. The heart of the soccer component is its behavioral control architecture, known as "Skills, Tactics, and Plays" (STP) [4]. Within STP, multi-agent coordination and team-based strategic decisions are performed by chosing from a collection of "plays". A play is defined as a list of role assignments for multiple robots. For example, in robot soccer, one might imagine an "offensive" play, which assigns most robots to be attackers, and some other robots to be defenders. The applicability of a particular play depends on whether that play's pre-conditions are currently met, given the value of some variables of the domain. If multiple plays are simultanously applicable then one of them is chosen probabilistically. The role assignment (e.g. which particular robots become an attacker) is then performed dynamically based on the robots' positions and availability. Individual robot roles are called tactics and are typically implemented as state-machines consisting of lower level navigational functions, called skills. Examples of soccer tactics include "Attacker", "Defender", and "Goalie".

The data used in our experiments originate from the performance of the CMDragons team at RoboCup 2007. The data was gathered over four round robin games against teams named "Wright Eagle", "RoboDragons", "B-

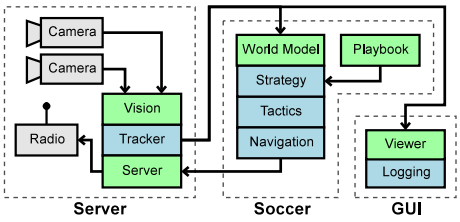

Fig. 2 The general architecture of the CMDragons offboard control software.

Smart", and "Botnia", followed by the quarter-final, semi-final, and final games against "Eagle Knights", "ZJUNlict", and "Plasma-Z", respectively. For each game, the system has generated a log file describing the game observations, as well as the CMDragon's internal system state. Log entries carry the currently executed play and tactic names of each robot and additional globally observed variables, such as positions, and velocities of the ball and robots. Furthermore, they include the status of so-called referee events which indicate the change of the global game status such as, e.g., a switch from a "game-stoppage" to a "kickoff". Log entries are recorded at 60Hz in unison with the system's main processing loop, which leads to a new log entry about every 17 milliseconds.

3 Process Mining

Process mining is a field of analysis techniques that focuses on mining behavioral aspects from log data. Since the mid-nineties several groups have been concentrating on the discovery of process models from event-based data. Process models are structures—usually directed graphs—that model behavior. In [3] an overview is given of the early work in this domain. Despite the many relations to some work discussed in the Machine Learning domain, there are also many differences as the targeted process models reside at the net level (e.g., Petri nets) rather than sequential or lower level representations (e.g., Markov chains, finite state machines, or regular expressions). Therefore, process mining needs to deal with various forms of concurrency.

Over the years, the notion of process mining has broadened to the general idea of extracting non-trivial and useful information from process execution logs [1]. For example, in the context of the robot soccer domain described in the previous section, the real-world process of interest is a robot soccer match (see Figure 3). Multiple robots interact with each other in the process of the game, and we are able to record information about their activities. These so-called "event logs" are the starting point for process mining techniques, and they can be used to derive and enrich higher-level models, or to compare

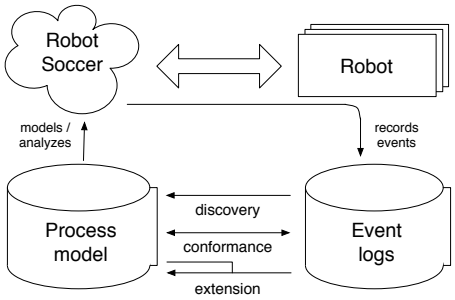

Fig. 3 Process mining in the context of robot soccer.

them to potentially existing models of behavior. Generally, we distinguish three classes of analysis:

Discovery *Extracting a new model from the event log.* In the context of robot soccer, for example, we may want to see process models of the overall team behavior, or of individual robot behavior.

Conformance *Comparing an existing model to the behavior observed in the log.* In the robot soccer example one could, for instance, compare the behavior of the opposing team with the own behavioral models or compare the observed behavior with some normative or descriptive model.

Extension *Projecting additional information onto an existing model.* Note that many aspects of a process (such as timing) cannot be known beforehand as they depend on the environment (for example, the opposing team). Hence, existing models can be enriched with derived information.

ProM [1] is a tool set that supports a wide variety of process mining techniques[1]. Furthermore, due to its plug-able architecture, it enables researchers to quickly implement new algorithms without spending any efforts on common tasks such as loading logs, displaying discovered models etc. In this study, we did not develop any custom analysis plug-in but used functionality that is readily available in ProM. Some sample analysis results are described in more detail in Section 5. However, in the next section we first take a closer look at the activity logs that form the input data for the process mining techniques.

4 Activity Logs

As described in Section 2, there are different levels of robot behavior. The high-level actions that we want to analyze are the *tactics*, which are the

[1] The software including source code and documentation can be freely downloaded from *prom.sf.net* (see also our website *processmining.org*).

different roles that a robot can have, such as "mark" or "position for pass". Since ProM uses the *M*ining *XML* (MXML) format to read event logs, the data collected by the CMDragons team needed to be converted into this format. The basic structure of MXML is very simple: a process log consists of a set of process instances, which in turn each contain a sequence of events. A process instance, also referred to as case, trace, or audit trail, is one particular realization of the process, while events correspond to concrete steps that are undertaken in the context of that process for the particular process instance. Furthermore, each event carries a timestamp and many contain additional data.

Note that already *during log conversion different views on the process can be taken*, since the definition of what is to be considered a process instance determines the scope of the process to be analyzed. For example, in the context of a robot soccer match one could be interested in the overall team formations over multiple games, as well as individual robot behaviors within a single game. Thus, different conversions can be leveraged to obtain different views on the same process. Using a custom plug-in of the ProM*import* framework [6] we have created both *individual robot logs* and *team logs*. Because the overall team behavior is globally orchestrated, we can easily get information about the current role of all the robots in the team at each point in time, and we consider a team formation by abstracting from which robot engages in which role. Thus, in a team log we see an action as a set of roles. Furthermore, we abstract from the goalie (as there is always one goalie).

An excerpt of an event in such a team log is shown in the following MXML log fragment. The event (`AuditTrailEntry`) was recorded in the context of the game against "Plasma-Z" (`ProcessInstance`) on 8 July 2007 at 12:57:18 according to US Eastern Time Zone (`Timestamp`), and refers to the team formation "Mark Mark Wall Wall" (`WorkflowModelElement`) while the ball was in possession of the CMDragons (`Data`) etc. The dots indicate that the log contains further process instances, and the process instance further events. Furthermore, the event contains more data attributes than is shown here.

```
<Process id="TeamLog" description="All matches together"> ...
<ProcessInstance id="4" description="Game PlasmaZ"> ...
<AuditTrailEntry>
<Data>
<Attribute name="ball_possession_ours">1</Attribute> ...
</Data>
<WorkflowModelElement>Mark Mark Wall Wall</WorkflowModelElement>
<EventType>complete</EventType>
<Timestamp>2007-07-08T18:57:18.709+02:00</Timestamp>
</AuditTrailEntry> ...
</ProcessInstance> ...
</Process>
```

The complete team log contains more than half a million (624,801) events that are distributed over 7 process instances. Because we are mainly interested in changes of behavior, we subsequently filter out repetitions of the

same team formation. This way, we yield a much more condensed log focusing only on state changes, while preserving information about the timing behavior through the time stamps. The filtered team log now contains only 2,183 events for all the 7 games, with the shortest process instance listing 63 and the longest 762 team formation changes. The same can be done for individual robot logs.

5 Analysis and Results

Due to space limits we can not go into detail about individual mining techniques, and only consider two illustrative examples of *discovery* (cf. Section 3), namely (a) process model discovery and (b) decision rule mining.

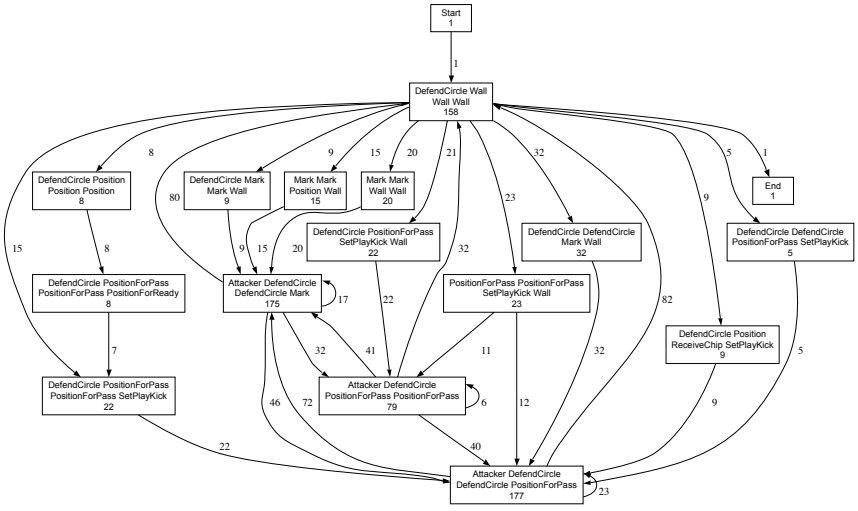

Fig. 4 Process model discovered by the Heuristics Miner based on the team log of the final game between CMDragons against "Plasma-Z". Each rectangle corresponds to a team formation of the CMDragons in that game (without the goalie).

(a) Based on the team log, we use the *Heuristics Miner* [10] to automatically construct a process model of the team formations in the final game against "Plasma-Z", which is depicted in Figure 4. It shows the causal dependencies between activities and provides an overview about the actual flow of the process, whereas each rectangle corresponds to a team formation of the CMDragons in that game (the numbers reflect the frequencies at which the team formation changes were observed). Now consider Figure 5, which was created from an individual robot log, and thus depicts the discovered process model of one individual robot in the same game. We can see that the robot

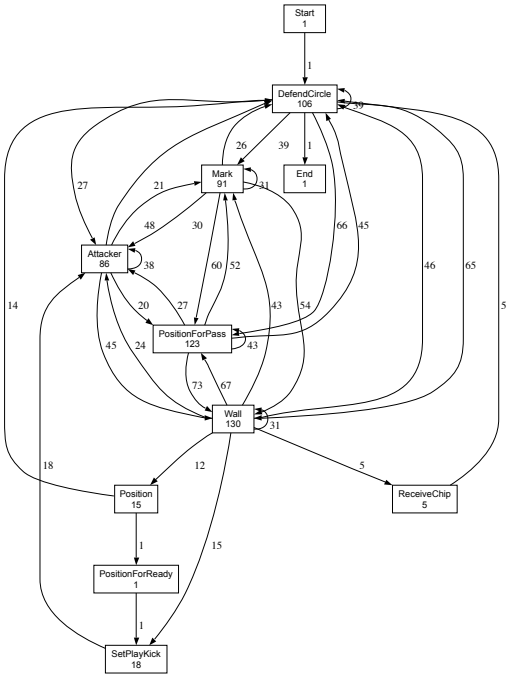

Fig. 5 Process model of one individual robot's behavior.

was frequently changing into the role "Wall" (130 times) and that it was, for example, frequently switching from the role "Wall" into "PositionForPass" (67 times) and back (73 times). Note that these models could now be further enhanced by making use of the time information in the log and visualizing the length of stay in each of these states by different colors.

(b) We apply the *Decision Miner* [8] in combination with the Weka machine learning workbench [11] to discover under which conditions which team formation is being chosen by the system. For this, we create a Petri net process model that represents the playbook of the soccer component (cf. Figure 2), which is partly shown in Figure 6, and we restrict the used data attributes to those used in the pre-conditions of the world model. As described earlier, the internal behavioral model assigns probabilities to the different plays, which are active as soon as their pre-conditions are fulfilled. Apart from these pre-conditions, every team formation is possible at each point in time (represented by "playing game" in Figure 6). Note that the internal model is more general than the discovered models in Figure 4 and Figure 5, which depict typical action flows more closely. Thus, the discovered models can provide real insight into the process as it took place, different games can be compared etc. It is interesting that we can discover the same or similar rules to a large extent. For example, the discovered rule '*their_kickoff => Mark Mark Wall*

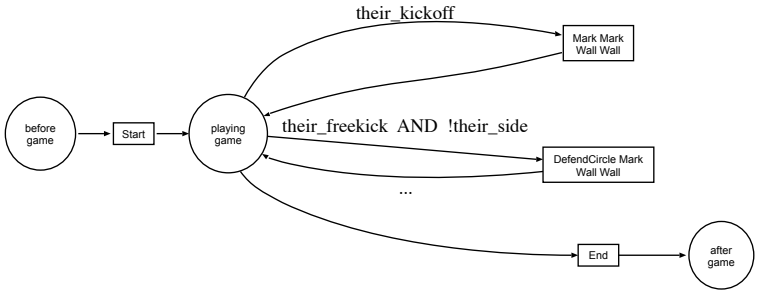

Fig. 6 Petri net process model reflecting the world model, where each of the team formations is possible at every point in time, if certain pre-conditions are met. The discovered rules (two examples are shown) largely coincide with the internal rules.

Wall' coincides with the original one and has both precision and recall of 1.0. The discovered rule '*their_freekick AND !their_side => DefendCircle Mark Wall Wall*' is slightly more general as the original rule was '*their_freekick AND our_side AND !our_corner*'. The overall accuracy of the discovered rules (using 10-fold cross validation over the whole team log) is between 63 and 74 % (depending on the classification algorithm). If there are mismatches, then this is mainly because some plays share the same pre-conditions and are then selected randomly. If we include additional attributes not directly used in the pre-conditions (e.g., referee status) the accuracy exceeds even 90 %. While it is less useful to re-discover one's own rules, this seems directly useful in the context of analyzing the opponent's behavior ("When do they react how?").

6 Conclusion

In this paper, we have shown the potential of structured log analysis to gain more high-level insight into the behavior of multi-robot systems. We have demonstrated based on the robot soccer domain that it is possible to apply process mining to multi-agent activity logs.

Many other techniques are applicable but have not been mentioned. For example, while the global observation system is very accurate and provides high-qualitative data, in turbulent phases of the game sometimes not all attributes can be reliably determined. Thus, it can happen that the ball possession is "ours" and "theirs" at the same time. Using linear temporal logic (LTL) one can check such pre-defined properties ('<>(*ball_possession_ours AND ball_possession_theirs*)') and further investigate those cases to improve the situation. Finally, custom implementations in the context of the ProM framework are possible. They have the advantage that they—while building

on common functionality—enable domain-aware and specialized solutions.

Acknowledgements. This research is supported by the IOP program of the Dutch Ministry of Economic Affairs. The authors also thank the CMDragons team for sharing their RoboCup log data, and Douglas Vail who helped us to create the ASCII logs used for conversion to the MXML format. Furthermore, we thank all ProM developers for their on-going work on process mining techniques. Special thanks to Christian W. Günther for NikeFS2, which makes it possible to work with large real-world logs, and for his GUI redesign, which continues to impress people and makes using ProM so much more fun.

References

1. W.M.P. van der Aalst, B.F. van Dongen, C.W. Günther, R.S. Mans, A.K. Alves de Medeiros, A. Rozinat, V. Rubin, M. Song, H.M.W. Verbeek, and A.J.M.M. Weijters. ProM 4.0: Comprehensive Support for Real Process Analysis. In J. Kleijn and A. Yakovlev, editors, *Application and Theory of Petri Nets and Other Models of Concurrency (ICATPN 2007)*, volume 4546 of *Lecture Notes in Computer Science*, pages 484–494. Springer-Verlag, Berlin, 2007.
2. W.M.P. van der Aalst, H.A. Reijers, A.J.M.M. Weijters, B.F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek. Business Process Mining: An Industrial Application. *Information Systems*, 32(5):713–732, 2007.
3. W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
4. B. Browning, J. Bruce, M. Bowling, and M. Veloso. Stp: Skills, tactics and plays for multi-robot control in adversarial environments. *IEEE Journal of Control and Systems Engineering*, 219:33–52, 2005.
5. J. Bruce, S. Zickler, M. Licitra, and M. Veloso. CMDragons 2007 Team Description. Technical report, Tech Report CMU-CS-07-173, Carnegie Mellon University, School of Computer Science, 2007.
6. C.W. Günther and W.M.P. van der Aalst. A Generic Import Framework for Process Event Logs. In J. Eder and S. Dustdar, editors, *Business Process Management Workshops, Workshop on Business Process Intelligence (BPI 2006)*, volume 4103 of *Lecture Notes in Computer Science*, pages 81–92. Springer-Verlag, Berlin, 2006.
7. K. Han and M. Veloso. Automated robot behavior recognition. 2008.
8. A. Rozinat and W.M.P. van der Aalst. Decision Mining in ProM. In S. Dustdar, J.L. Faideiro, and A. Sheth, editors, *International Conference on Business Process Management (BPM 2006)*, volume 4102 of *Lecture Notes in Computer Science*, pages 420–425. Springer-Verlag, Berlin, 2006.
9. D. Vail, M. Veloso, and J. Lafferty. Conditional random fields for activity recognition. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM.
10. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
11. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.