The Glee Didn't Last Long: How Continuous Integration Helps Revealing Fundamental Flaws in Team-Play

Thomas Gabel and Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|godehardt}@fb2.fra-uas.de

Abstract. We present a continuous integration and deployment (CI/CD) framework for Soccer Simulation 2D. On the one hand, we aim to share that system publicly with the community and, therefore, describe its components and their composition. On the other hand, we relate statistics produced by that system to concepts from reliability theory, which facilitates the derivation of reliabilty metrics for assessing a team's defensive performance. In line with that, we present the results of a comprehensive case study in which the CI framework was utilized to find critical game situations, uncover weaknesses in the playing behavior of a soccer simulation team, and implement and evaluate appropriate counter measures.

1 Introduction

The traditional approach to developing and debugging agents in Soccer Simulation 2D – as well as to agent-based programming in general – has often been to program and improve components of their behavior iteratively and episodically: A human developer observes some sub-optimal playing behavior, infers the potential causes in the source code, programs some counter measures or improved versions of existing algorithms, evaluates them by a handful of test situations or matches, and when having approved the changes made, commits them to some code repository. Over the years, a number of approaches, tools, and methodologies were established to improve and support the sketched workflow. Layered disclosure as introduced by Carnegie Mellon's soccer simulation team [13] was one of the successful early approaches frequently adopted by other teams to allow for a goal-directed inspection of an agent's internals boosting the capabilities of developers in debugging agent behavior and in learning from logged data.

Over the years, the official tools maintained by the 2D Soccer Simulation community, most prominently the monitor and log player tools [1], became mature and incorporated more and more facilities that allowed for enhanced match analysis. Likewise, many active soccer simulation 2D teams have developed – and often released to the public – their own tools to support development, debugging, and agent analysis. Those include the set-play mechanism by FC Portugal

[9], the Soccer Reporter and itas2d-window tools by ITAndroids [16], or the AutoTest2D scripts by WrightEagle [2].

While the general procedure discussed above is suitable for prototyping, correcting (obvious) errors, and often implies some (programming) fun, it quickly comes to its limits when questioning the impact of changes to the playing performance of a team. Due to the inherent stochasticity of soccer simulation and the enormous heterogeneity of opponent teams' playing strategies, it is very challenging to foresee the exact influence of specific code changes at large. At this point, evaluations by playing a larger number of games and drawing statistically reliable conclusions from them comes in. While this idea is not new – several robotic soccer simulation teams have published related approaches (cf. Section 2.3) –, the system that we outline in the next section and that we share as open source with the community has the property of implementing the continuous integration and deployment (CI/CD) approach. It facilitates the automatic generation of numerous statistics that allow developers to immediately spot and further analyze weaknesses in a team's performance. Moreover, in Section 3, we propose casting the problem of defending in robotic soccer as a reliability problem where conceding a goal is considered a failure. Utilizing concepts and approaches from the field of reliability theory and engineering, we then show in Section 4 how the outputs created by our CI/CD system can be further processed and deployed to obtain reliability metrics. This enabled us to discover and repair flaws in our (FRA-UNIted) robotic soccer team's tactics.

2 A Continuous Integration Framework for Soccer Simulation 2D

The FRA-UNIted team [4] has an extensive history resulting in a sizable C++ codebase, posing challenges for making confident changes without adequate safeguards. Implementing a continuous integration and deployment strategy can mitigate these risks. However, evaluating the team's performance based on a few games is insufficient. As argued in [6], conducting approximately a thousand games is most often sufficient to obtain a reliable assessment of whether recent changes have had a positive or negative impact.

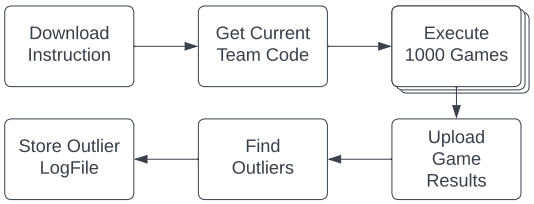

Fig. 1. General Flow of Execution Done Every Night

In order to implement this the flow in Figure 1 is executed in parallel on 20 computers. This process usually starts at night, when students have left the university. Each one of these computers will first download the current set of instructions, including specific teams for left and right side (dubbed "home" and "away" team, subsequently), or the current team code from git, respectively. In addition, the number of matches and some configuration options can be specified.

Thereafter the envisaged number of games is executed in parallel using the HLM tool — "Hechi's League Manager", initially developed by team KickOff-TUG's Hechenblaickner [7] and used as official league manager at RoboCup events for years. As all the produced log files would be way to much to keep, only a condensed version is upload to the back-end, where finally outliers in this nightly set of games are identified. The main idea is, that outliers are probably either exceptionally good or bad, and might be helpful to replay and beneficial for further analysis in order to improve our team code.

2.1 System Architecture

Besides the general flow of steps, the overall architecture will be explained now. Figure 2 shows both, components involved and the detailed flow of activities that are taking place. The whole process is initiated by all clients by downloading the latest instructions from the back-end at 8 pm. Numbers show the order of activities ending in storing outlier information. In addition, a web front-end is provided to analyze different statistics. Small parts of the front-end are shown in Figure 3 and will be discussed in Section 4. The front-end also enables the user to adjust the instructions to run at night (team setup, number of games etc.).

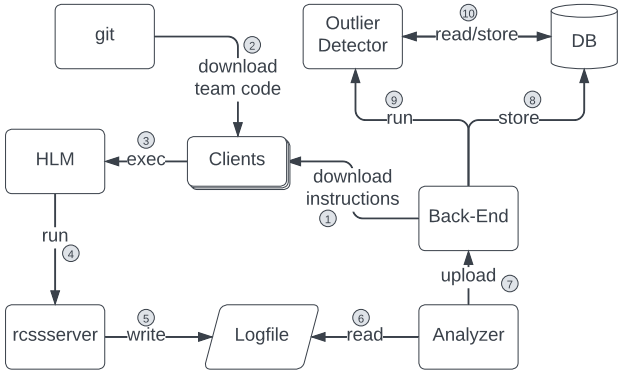

Fig. 2. Mixture of CI/CD Architecture and Activity Diagram

Docker is used on the clients as well as the server side. On the server, a Docker compose is running containing the back-end (together with the compiled front-end), a MongoDB and our outlier detector. The clients also utilize Docker

to have a reliable setup and to not interfere with the specific software setup of the client. That sets the stage every night for a clean environment. This is not the same setup, that is run at the world championships, where each and every agent is running in a separate container. Instead all 22 agents and the Soccer Server run jointly within *one* container.

2.2 Open-Source Release

The source code of our CI/CD setup is released as open source in a collection of four GitHub repositories[1]. As described above, these repositories together achieve our night CI/CD setup to run and analyze a thousand games a night.

2.3 Related Work

Approaches similar to the one we have sketched above have already been taken by other Soccer Simulation 2D teams. Team Helios [18] uses a performance assessment system, which leverages SlackBot, Amazon S3, and Google Sheets. Its users are empowered to initiate tasks via SlackBot, specifying branch, opponent details, and the desired number of games. A server orchestrates the allocation of client machines to tasks based on CPU load, abstaining from assigning tasks to heavily burdened computers. Subsequently, client machines execute designated games, process resultant log files into CSV format, and upload them to shared storage platforms such as Dropbox or Amazon S3. These CSV files are then consolidated within a Google Sheet, facilitating user access for performance evaluation purposes. Team Fractals [12] also performed extensive evaluations of its team and variations thereof in the context of a guided self-organization methodology to optimize team performance amidst the fluctuations inherent in the non-deterministic RoboCup simulation environment. The authors developed and analyzed a newly proposed method called Dynamic Constraint Annealing whose execution took approximately three days of computation on a high-performance cluster running 100 parallel games in the RoboCup Soccer Server's synchronization mode, with each game taking approximately 2 minutes. Team Cyrus has developed the AutoTune2D tool [15] which is an extension of the older AutoTest2D system [2] and which allows researchers to fine-tune parameters that influence a team's playing strategy. Finally, it should be noted that CI/CD approaches are not unique to the simulation league, but have been applied in robot-based RoboCup leagues as well, for example in the Standard Platform League [3].

3 Basics of Reliability Theory

In the subsequent section, we adopt a reliability engineering approach to analyzing the performance of a team (more specifically, the performance of a team's

[1] https://github.com/godehardt/fraunited-backend,
https://github.com/godehardt/fraunited-frontend,
https://github.com/godehardt/fraunited-outlier-detector and
https://github.com/godehardt/fraunited-ci

defense). For that reason, we need to briefly introduce some basics of reliability theory here, also pointing to [8] and [11] for the interested reader.

Reliability theory is the branch of statistics that focuses on the application of probability theory in an engineering context to the modeling of the occurrence of events such as failures and the prediction of success probabilities. In related research disciplines the same mathematical concepts are applied under the names survival analysis (biostatistics), event history analysis (sociology), or duration analysis (economics). Events of interest may, thus, include the death of a living being, the failure of a mechanical system, or – as in the case of robotic soccer – the conceding of a goal (which somehow represents the failure of a team's defense).

Random Time to Failure In terms of reliability, the time to failure is modeled as a random variable T where $P(t \leq T < t + \Delta t)$ denotes the probability that the time to failure is in the interval $[t, t + \Delta)$. Accordingly, the failure probability density function f and distribution function F are defined as usually in probability theory as

$$f(t) = \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t)}{\Delta t} \text{ and } F(t) = \int_0^t f(x)dx \tag{1}$$

where the latter also represents the probability that a failure occurs by time t, which in the soccer domain boils down to the probability that a goal is conceded by t.

Reliability Function The complement to F is represented by the chance of being successful at least until t, e.g. keeping a clean sheet in soccer. This complement is called the *reliability function* R and denotes the probability that the time to failure is larger than t (i.e. $T > t$):

$$R(t) = P(T > t) = 1 - F(t) = \int_t^\infty f(x)dx. \tag{2}$$

Failure Rate and Hazard Rate Interestingly, research in reliability theory has shown that the density function f is not very useful in practice which is why the *failure rate* function λ is derived. It represents the ratio of the probability that a failure occurs per unit time within some interval $[t, t + \Delta t)$ *and* the probability that the system has survived faultlessly up to t:

$$\lambda(t) = \frac{P(t \leq T < t + \Delta t)}{\Delta t} \cdot \frac{1}{P(T > t)} = \frac{F(t + \Delta t) - F(t)}{\Delta t \cdot R(t)}.$$

The limit of the failure rate for $t \to 0$ is called the *hazard rate* h, for which we obtain from Equation 1 that

$$h(t) = \lim_{\Delta t \to 0} \frac{F(t + \Delta t) - F(t)}{\Delta t \cdot R(t)} = \frac{f(t)}{R(t)} = \lim_{\Delta t \to 0} \frac{-(R(t + \Delta t) - R(t))}{\Delta t \cdot R(t)} = -\frac{d}{dt} ln(R(t)). \tag{3}$$

In our notion, the hazard rate is the instantaneous rate of conceding a goal at time t given that the team has not received a goal against up to t.

Goal Rate Although there is some minor difference in the definitions of hazard and failure rate, they are often used interchangeably in the literature. Therefore, we will subsume them, calling it the (opponent) *goal rate* $g(t)$, subsequently. The goal rate will, in general, change over the course of a match, varying depending on play modes, ball position on the pitch, or playing strategies of the teams. In the next section, we will inspect the goal rate with the help of the presented CI system for a number of specific match situations.

4 Case Study

The case study delineated in the following is meant to showcase the usefulness of the presented CI/CD system and, beyond that, show how ideas from reliability theory can well be applied in the domain of robotic soccer.

4.1 The Changeover Anomaly

When applying the CI system regularly in the course of preparing for the RoboCup 2023 tournament, we soon discovered an anomaly within the visualized results and statistics produced by the React web application that is part of the CI system's back-end. Among the many statistical visualizations [17] is a goal histogram which consistently showed a peak of goals conceded right after the half-time break. Figure 3 (left) provides an impression of that anomaly for a series of 3000 matches of our team, FRA-UNIted, against the 2022 champion team Helios [18]. Using a bin size of 100 time steps, the summed amount of goals seems to be nearly constant on average throughout the 6000 time steps of a game, except for the beginning where both teams score less which can easily be explained by missing knowledge about heterogeneous player types [10] which are typically inferred after a few hundred time steps. Quite remarkably, however, the curve of goals conceded by our team (blue in the left chart of Figure 3) contains an outlier right after the changeover with almost three times as many goals per bin as during the rest of the match.

We first speculated that this may be caused by defective tactical assignments of players to to-be-marked opponents which are determined by the coach and communicated to players using the Coach Language [14] or by a (partial) reset of those assignments at half-time. Although this calculation and ensuing interaction between agents had proved to be fragile in previous times (as delineated in [5]), we could verify that this is not the root of the anomaly on this occasion.

A second conjecture, which we also soon discarded, was that there might be an issue in the way the side changeover at half-time is realized in our agents' world models. In this context, we once again employed the CI system to generate another series of (9000) matches against the same opponent team, however, with sides flipped, i.e. the fixture was now Helios vs. FRA-UNIted. While the average scores of both series did not show a significant change (0.59 : 1.83 for the first series, and 1.77 : 0.59 for the second one), the distribution of goals along the time looked quite different as shown in the right part of Figure 3. The peak at

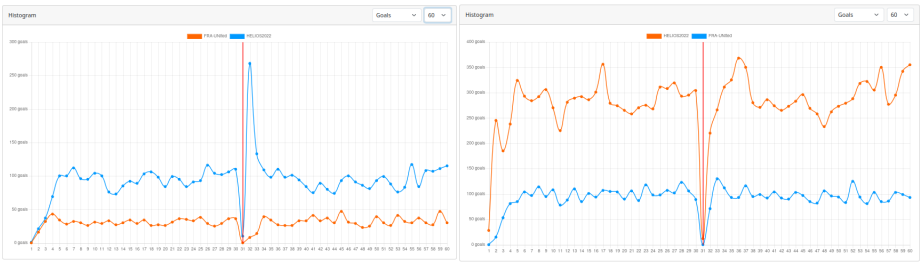

Fig. 3. Original Screenshots from the CI System's React Web Application Visualizing Game Series Statistics Produced by its Python-Based Statistics Tool. Here, the total time of a match is binned using 60 bins, i.e. a window size of 100 time steps. Left: Goal histogram for 3000 matches with FRA-UNIted as the home team, shown in orange, and Helios as the away team, shown in blue. Apparently, something peculiar is going on at half-time breaks. Right: Goal histogram for 9000 matches with sides swapped, i.e. Helios is now the home team (its goals in orange) and FRA-UNIted the away team (blue). Obviously, the goal distribution seems to have changed when swapping sides. Note: Both figures show total number of goals summed over different numbers of matches (3000 vs. 9000). Therefore, the ordinates are scaled differently. For the mid and end of each half-time not much has changed (e.g. about 100 goals for Helios in bin 9 (left) and 300 goals in bin 9 (right) from three times the number of games), but for the beginnings of the halfs the curves differ a lot.

half-time disappeared, whereas the slope for the team playing from left to right (home team, orange curve, i.e. Helios in the 2nd series) at the very beginning of the match was much steeper.

The Soccer Server lets always start the game with a kick-off for the home team, whereas the away team kicks off to the second half of the match after the half-time break. Given that in the first series the considered opponent Helios was always the away team and, thus, had a kick-off in each game at half-time ($t = 3000$) and in the second series at game start ($t = 0$), we concluded that the initially mentioned peak in goals against FRA-UNIted must be attributed to a poorly-developed and suboptimal behavior of our players during opponent kick-off situations.

4.2 The Glee Didn't Last Long: Conceding Goals After Scoring

According to soccer rules, a single kick-off for the opponent team is unavoidable in any match (either at the beginning or at half-time). Yet, another kick-off is awarded to the opponent after each goal that our team has shot. The conclusion from the previous section, hence, also elicited a profound realization: The joy after having scored a goal may, after all, not last long, if the opponent's probability of scoring a goal right after the immediately following kick-off is high. In order to thoroughly examine this question – and in so doing to confirm that the delineated weakness is really due to kick-offs – we once again utilized the data produced by the CI system discussed above.

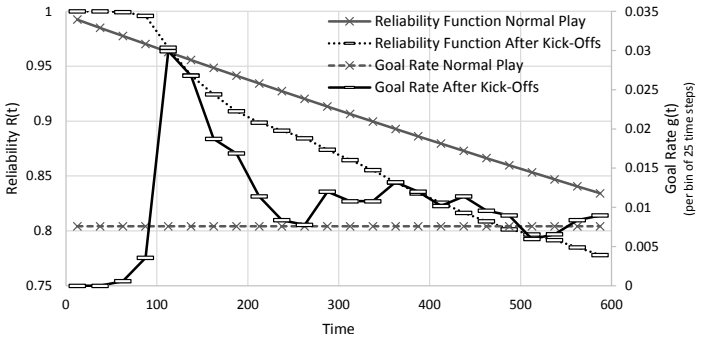

Fig. 4. What Happens After Having Scored: The chart focuses on the 600 time steps after our team has scored a goal and the immediately following opponent kick-off has taken place, aggregating data from 3000 matches. After a little more than 100 time steps, the goal rate of the opponent increases severely while the reliability of our defense drops sharply. This is in stark contrast to the overall "normal play", i.e. the average over all game situations over all 3000 matches, which are drawn in gray color with cross marks, where the goal rate is approximately constant.

The back-end of the CI/CD system we are releasing here also contains a Python-based analyzing tool which is utilized to generate match statistics in JSON format stored in a local database (cf. Section 2.1). We used that data and applied various approaches from reliability theory (cf. Section 3). Knowing that a kick-off takes place after goals, we selected the subset of match situations unfolding after our team's goals, excluding, however, those that took place near the end of a game or shortly before the half-time break in order to avoid the well-known censoring problem in reliability theory [8].

Figure 4 plots the reliability function $R(t)$ (cf. Equation 1) for the time after our team has scored a goal (i.e. goal for FRA-UNIted at $t_0 = 0$) where, as discussed in Section 3, R refers to the probability of surviving without conceding a goal till time t ($t > t_0$). More interestingly, that chart also contains the hazard rate function $h(t)$, which we refer to as goal rate $g(t)$, that denotes the instantaneous rate of receiving a goal at $t > t_0$. The data is based on 3000 matches with about 1600 goals for FRA-UNIted and immediately following kick-offs for Helios and has been binned with a window size of 25 time steps to produce more readable figures. For comparison, the reliability function and goal rate as they emerge on average during "normal play" are plotted as well. While the goal rate in normal play is constant at $g(t) = 0.0076$ (so many goals are hit within a time window, i.e. within 25 time steps, on average) it peaks with a value almost four times as high (0.03) after circa 100-125 time steps after a kick-off (immediately after a goal by us) has taken place. Note that these numbers exclude kick-offs that took place regularly, i.e. at the start of a half-time. In a separate analysis (not plotted for readability) we confirmed that the reliability function and goal rate for those "regular" kick-offs at half-time or game start are nearly identical.

4.3 Eradicating the Kick-Off Weakness

After acknowledging that our defense's reliability right after having scored a goal, i.e. during opponent kick-offs, is clearly below average, we started working on improvements. The details of the flaws found and belonging corrections are beyond the scope of this paper and can be read in a future team description paper [4]. In brief, we noticed subpar distances between our defenders and their to-be-marked opponent players in conjunction with at times undetected role changes by opponent attackers, often resulting in attackers overrunning our defense line.

Most of all, however, we had to admit that opponent kick-off situations were, in general, far less than perfect tested by our (human) team members. This is because human-based development and testing (even when improved by means of layered disclosure, as described in Section 1) focuses very often on *partial* matches with the team under development as home team (i.e. kicking off at game start) and some arbitrary (last year's) opponent binary as away team (thus, not kicking off at game start). Such development test games are often interrupted by the developer as soon as a goal is shot by either team or some other interesting event occurs that seems worth analyzing in the log files. This way, however, often partial matches with a duration of less than a half-time are inspected which do not include any opponent kick-off, which is why those events are extremely underrepresented and flaws are hardly recognized.

Although this delineation is episodical for our team we are, on the one hand, convinced that it holds for other soccer simulation 2D teams as well. On the other hand, the described experience makes a strong case for the introduced CI system which automates much of the testing and assessment work and, in so doing, aims at preventing the mentioned problem of underrepresenting certain match situations.

Results The effectiveness of the improvements can be read from Figure 5. Utilizing the CI system once again, we ran another series of 3400 test matches in the same set-up as before, this time, however, employing our mentioned kick-off-related modifications. Again, our focus was on the reliability (top left) of our defense in preventing opponent goals as well as on the goal rate (top right) that emerges after our team's goals and the immediately following opponent kick-off. Apparently, although still worse than during normal play, the defense reliability (top left) after opponent kick-offs could be improved significantly when compared to our team's original (released 2023 binary) version. This is also reflected in the goal rate arising after our goals and the subsequent opponent kick-offs (top right part of Figure 5), whose peak at $t \in [100, 125]$ could be mitigated clearly.

The top right diagram contains two further curves which refer to the *cumulated* goal rate $G(t)$, for which Equation 3 yields

$$G(t) = \int_0^t g(x)dx = \int_0^t \frac{f(x)}{R(x)}dx = -ln(R(t)).$$

The curves put this value $G(t)$ in relation to the cumulated goal rate during normal play (plotted on the secondary ordinate, with 100% representing normal

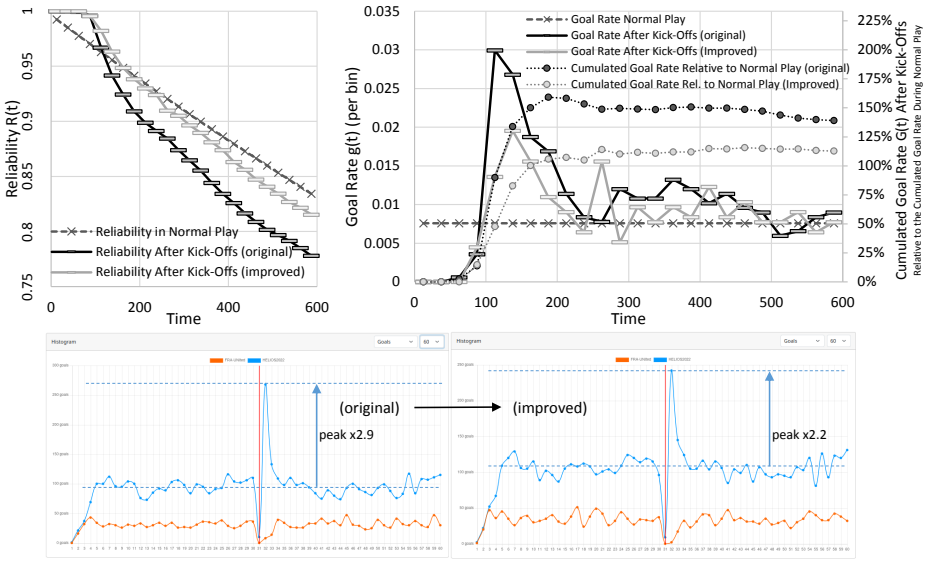

Fig. 5. Efficacy of Counter Measures Against the Detected Kick-Off Weakness: Both charts in the top row contrast reliability and (cumulated) goal rate before ("original") and after ("improved") having implemented counter measures. The goal histogram charts in the bottom row were generated with the presented CI system.

play). Apparently, our team's cumulated risk of receiving a goal (within 600 time steps) after having shot a goal used to be 39.0% higher than during normal play, whereas it is only 12.8% higher after our modifications, hence proving their effectiveness using the CI system.

Returning to the changeover anomaly, the two bottom charts of Figure 5 which were automatically rendered by the CI system, prove that our modification had the same positive expectable impact on what happens after the opponent's regular kick-offs at half time. The magnitude of the goal peak at half-time relative to the general distribution of goals could be reduced significantly. Note that these histogram charts plot total numbers originating from different amounts of games, therefore being scaled slightly differently.

Discussion The CI system is composed as a loose coupling of a set of components and, therefore, provides the flexibility to shift the focus to other game events as well. For example, we have done a brief analysis on the vulnerability of our team's playing behavior after opponent free kicks and corner kicks. As can be seen in Figure 6, we obtain reliability and goal rate progressions that are worth considering. Obviously, opponent free kicks seem to exert only moderate amount of extra danger to our defense (especially between 200 and 300 time steps after the free kick was taken), By contrast, the observed goal rates after corner kicks by the opponent team (with ca. 0.077 per bin at $t \in [100, 125]$ almost three times as high as after kick-offs, cf. Figure 4) seem to leave room for improvement.

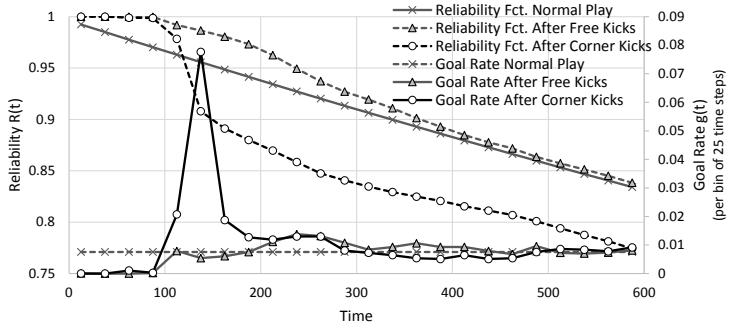

Fig. 6. Reliability and Goal Rate After Further Game Events: This figure contrasts the average "normal play" defense reliability and opponent goal rate with those that are to be observed after opponent *free kicks* and after opponent *corner kicks*. The insight is, that our team's reliability is even higher than average after free kicks indicating well-developed defense behavior. The opposite is true for opponent corners for which our team's behavior seems to be poorly adapted, as indicated by a goal rate that clearly stands out for times in [125,175] steps after taking the corner kick.

Similarly we might extend the breadth of the analyses by focusing on, for instance, our team's standard situations or on arbitrary game situations, like "ball has left the penalty area" or "an opponent self-pass has been intercepted" to analyze the corresponding subsequent progress of the match. Moreover, so far our focus has been on ultimate outcomes, i.e. goals (received or shot), but, ultimately, other game events and statistics like "share of successful passes" or "ball possession rate" might be be considered just as well.

5 Conclusion

We have presented the continuous integration and deployment system that we developed and maintain as part of our activities in the realm of robotic soccer simulation. The system enables the automatic conduction of large series of matches against definable opponents upon any code change conducted, is able to filter out games based on outlier rules, and facilitates the derivation of game statistics that can guide the developers in improving the team. We released the system under MIT licence on Github, making it available to the 2D community. In order to demonstrate the usability of the system, we also presented a case study where that system has been used end-to-end. It helped us uncover some awkward weaknesses in our team's playing strategy that occurred immediately after scoring a goal. Using concepts from reliability theory, it additionally assisted in analyzing the issue in depth and in aiming to ameliorate it.

Acknowledgements: This research has been supported by the German Federal Ministry of Education and Research under grant number 13FH027Kl1.

References

1. Akiyama, H., Zare, N.: The RoboCup Soccer Simulator (Last accessed 03/2024), https://github.com/rcsoccersim, Soccer Server, Soccer Monitor, Logplayer
2. Bai, A., Lu, G., Zhang, H., Chen, X.: WrightEagle 2D Soccer Simulation Team Description 2011 (2011), Supplement to RoboCup 2011: Robot World Cup XV
3. Estivill-Castro, V., Hexel, R., Lusty, C.: Continuous Integration for Testing Full Robotic Behaviours in a GUI-stripped Simulation. In: CEUR Workshop Proceedings of MODELS 2018. vol. 2245, pp. 453–464. Copenhagen, Denmark (2018)
4. Gabel, T., Eren, B., Godehardt, E.: FRA-UNIted – Team Description 2024 (To appear), Supplement to RoboCup 2024: Robot Soccer World Cup XXVI
5. Gabel, T., Riedmiller, M.: On Progress in RoboCup: The Simulation League Showcase. In: J. Ruiz-del-Solar, E. Chown, P. Plöger, editors, RoboCup 2010: Robot Soccer World Cup XIV, LNCS. pp. 36–47. Springer, Singapore (2010)
6. Godehardt, E., Allani, M., Vieth, A., Gabel, T.: 1001 Games a Night – Continuous Evaluation of an Intelligent Multi-Agent Based System. In: Proceedings of the 8th International Congress on Information and Communication Technology (ICICT 2023). pp. 193–208. London, United Kingdom (2023)
7. Gspandl, S., Arnus, W., Gebetsberger, J., Gsenger, G., Hechenblaickner, A., Reip, M., Wagner, C., Wolfram, M., Zehentner, C.: KickOffTUG – Team Description Paper 2010 (2010), Supplement to RoboCup 2010: Robot Soccer World Cup XXII
8. Lyu, M.: Handbook of Software Reliability Engineering. McGraw-Hill (1995)
9. Mota, L., Reis, L.: Setplays: Achieving Coordination by the Appropriate Use of Arbitrary Pre-Defined Flexible Plans and Inter-Robot Communication. In: Proceedings of the 1st International Conference on Robot Communication and Coordination (ROBOCOMM). vol. 318, pp. 1–13. ICST/ACM, Athens, Greece (2007)
10. Noda, I.: Soccer Server: A Simulator of RoboCup. In: Proceedings of the AI Symposium 1995. pp. 29–34. Japanese Society for Artificial Intelligence (1995)
11. O'Connor, P.: Practical Reliability Engineering. Wiley (2012)
12. Prokopenko, M., Wang, P.: Fractals2019: Combinatorial Optimisation with Dynamic Constraint Annealing. In: RoboCup 2019: Robot World Cup XXIII. pp. 616–630. Springer, Sydney, Australia (2020)
13. Riley, P., Stone, P., Veloso, M.: Layered disclosure: Revealing agents' internals. In: Castelfranchi, C., Lespérance, Y. (eds.) Intelligent Agents VII Agent Theories Architectures and Languages. pp. 61–72. Springer, Berlin, Heidelberg (2001)
14. Riley, P., Veloso, M., Kaminka, G.: Towards Any-Team Coaching in Adversarial Domains. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002). pp. 1145–1146. Springer, Bologna, Italy (2002)
15. Sayareh, A., Zare, N., Amini, O., Firouzkouhi, A., Sarvmaili, M., Matwin, S., Soares, A.: Observation Denoising in CYRUS Soccer Simulation 2D Team For RoboCup 2023 (2023), Supplement to RoboCup 2023: Robot World Cup XXV
16. Vasconcelos, D., Gottschild, G., Maximo, M., Farias, N., Almeida, V., Araujo, V., Araujo, Y.: ITAndroids 2D Soccer Simulation Team Description Paper 2022 (2022), Supplement to RoboCup 2022: Robot Soccer World Cup XXIV
17. Vieth, A.: A Continuous Integration System with Diversified Opponents and Dynamic Team Configurations for RoboCup 2D. Master's thesis, Frankfurt University of Applied Sciences (2022)
18. Yamaguchi, M., Kuga, R., Omori, H., Fukushima, T., Nakashima, T., Akiyama, H.: HELIOS2021: Team Description Paper (2021), Supplement to RoboCup 2021: Robot Soccer World Cup XXV