Performance Measurement and Software QA in the RoboCup Standard Platform League: A qualitative interview study

Thomas Klute

Robotics Research Institute, Section Information Technology,
TU Dortmund University, 44227 Dortmund, Germany
thomas.klute@tu-dortmund.de
https://www.irf.tu-dortmund.de/

Abstract. This study investigates how teams in the RoboCup Standard Platform League (SPL) manage software quality assurance (QA) and performance measurement, both of which are critical to achieving RoboCup's ambitious goals. Through qualitative interviews with eight SPL teams, we analyzed their practices, challenges, and strategies for balancing innovation, education, and competition.
The findings reveal that manual QA approaches, such as expert judgment during test games, are widely employed and remain a cornerstone even for the league's top-performing teams. Only a minority of teams have implemented more structured and automated mechanisms akin to those used in software engineering. Furthermore, almost none of the teams engage in performance measurement beyond manual assessment through observation. We explore potential explanations for these findings and conclude that there is plenty of room for improvement towards implementing more structured and automated approaches to quality assurance and performance measurement. This study highlights the potential for league-wide initiatives to promote standardized performance metrics and tools that enable teams to adopt more structured QA practices. By fostering automation and consistency, RoboCup SPL can better support teams in achieving both educational objectives and competitive excellence, paving the way toward the league's long-term vision.
These results hold practical significance for shaping the roadmap of RoboCup and offer valuable insights for establishing team-based quality assurance and performance measurement practices.

Keywords: RoboCup SPL · Performance measurement · QA · Software testing

1 Introduction

The RoboCup initiative, founded in 1997, provides a unique platform for research and education in robotics [1] with the ambitious goal of defeating a human soccer team with a team of autonomous robots by 2050 [2]. Halfway towards that goal, tremendous progress has been made, starting with simulated and wheeled robots

[3], then using the Sony Aibo dogs [4], and advancing to humanoid robots capable of demonstrating basic skills successfully, including elements of team play [5]. Within this ambitious framework, the Standard Platform League (SPL) serves as a testbed for innovation, offering teams a standardized hardware platform and focusing on software development. There remain 25 years for research and development work, advancing towards strategic team play and achieving the goal of competing with human football players. Achieving this goal requires continuous progress and improvement within the RoboCup leagues and heavily depends on research and development among the teams. Simultaneously, RoboCup has served as an educational field for students in robotics since its inception [6]. New students enter this complex field each year, joining RoboCup teams and striving to engineer high-quality robotic systems.

Balancing student education with the demands of creating top-tier robots poses significant challenges. RoboCup teams must maintain the quality and performance of their already achieved skills at an appropriate level while focusing on more recent research areas that include more sophisticated team behavior and tactical components of the game. This study explores how SPL teams navigate these challenges, particularly the strategies employed to maintain and improve software quality and performance. By investigating the existing practices and identifying areas for improvement, this research aims to provide insights that can guide both teams and league organizers in fostering more systematic and effective approaches to QA and performance measurement.

2 Background

RoboCup serves as a hub for robotics research, aiming to advance robotic skills for the broader research community. Additionally, it plays a significant role in student education, inspiring young people to engage with robotics and push the boundaries of scientific knowledge. Most RoboCup teams comprise both researchers and students, striving for top scores in their respective leagues. However, robot hardware and the control software required for safe and effective robot operation pose significant challenges. Numerous modules must function seamlessly within defined parameters to enable complex behaviors necessary for tasks, all while ensuring the safety of both robots and potential human players. Safety requirements have already been specified for industrial robots in ISO 10218 [7] and cooperative industrial robots in ISO 15066 [8], but no such specifications exist for soccer-playing robots in environments shared with human players.

While tasks such as humanoid robot locomotion, image processing, and position tracking are challenging yet solvable, these will soon become fundamental skills for all RoboCup teams. As we approach 2050, the focus must shift towards faster, more agile movements, sophisticated ball handling, improved passing and more vigorous kicking, audiovisual communication, and coordinated teamwork among multiple robots - all within strict safety constraints to ensure the well-being of all participants.

We notice that existing teams and newcomers struggle to keep pace with the evolving complexity of the league, prompting questions about better strategies to help them maintain or enhance the quality of their software over time. Given RoboCup's emphasis on student education, where many participants start with little prior robotics knowledge, it is crucial to understand how teams can remain competitive while managing these challenges.

3 Methods

3.1 Aim

The study aims to collect examples of the challenges faced by SPL RoboCup teams, their general approach to software development, and the methods they use for quality assurance and performance measurement.

3.2 Research Questions

Our primary research questions are:

- Who is responsible for quality assurance within RoboCup SPL teams, on what code base do the teams work and how successful are they?
- What approaches do RoboCup SPL teams use to maintain and improve their playing strength, and how do these approaches account for the varying levels of expertise among team members?
- What tools and processes are utilized by SPL teams to measure and improve their performance on the field?
- To what extent do SPL teams adopt structured software development life cycles (SDLC) and integrate automated QA tools into their workflows?
- What are the main challenges faced by SPL teams in maintaining software quality and competitive performance, and what solutions have they proposed or implemented?
- How can the league's structure or guidelines support teams in improving their QA and performance measurement practices?

3.3 Design

To gather accurate responses to our research questions, we initially asked the teams, who among them is responsible for quality control and performance benchmarking. If this role was not explicitly filled by a team member, we asked the team leader or the person responsible for this process.

The interview was structured into sections covering:

- Personal and scientific background of the participants, their role in the RoboCup team, and related research work.
- Team details, including tournament placements, code base, research activities, simulator usage, and its perceived accuracy.

- Approaches to quality assurance and performance measurement, including test coverage, structured software development cycle, and continuous integration (CI) pipelines.
- Future steps, including proposed improvements for the team and league.
- Feedback on a potential automated testing approach tailored to RoboCup's specific requirements.

3.4 Participants

For this study, we conducted interviews with relevant people from eight RoboCup teams (B-Human, Bembelbots, Dutch Nao, HTWK, HULKs, Naova, rUNSWift, and our team, the Nao Devils), We primarily focused on interviewing successful teams within the SPL, as these teams demonstrate the ability to achieve consistently strong results in competitions. By analyzing their QA and performance measurement approaches, we aimed to identify effective practices that contribute to their sustained success. Two main factors influenced the selection process: the proven competitiveness of the teams over recent years and the availability of relevant team members during the tournaments. While we prioritized teams that had consistently performed at a high level, the practical constraints of time and team member availability limited the number of interviews during the venues. Despite practical constraints, the study includes insights from top-performing teams, offering a robust basis for analysis. The selection included the two best teams over the last years (B-Human and HTWK) and four teams often ranked in the top eight of competitions (Bembelbots, HULKs, rUNSWift, and our team, the Nao Devils) over the last years.

3.5 Data collection and analysis

Data were collected through 60- to 90-minute interviews at the German Open Replacement Event (GORE) in Hamburg and during RoboCup 2023 in Bordeaux. Additional data were gathered during RoboCup 2024 in Eindhoven. The interviewer took field notes, which were supplemented and cross-checked with information from team websites and official RoboCup sources to validate details such as code base size and tournament rankings. These raw data were manually processed later to extract and summarize the relevant facts and statements.

4 Rigour

To ensure trustworthiness according to Lincoln and Guba [9], credibility was ensured by discussing the findings and their derivations with other researchers at our institute. The other researchers were familiar with the RoboCup SPL league. Confirmability was achieved through the description of the analysis process, the archiving and publication of the anonymized raw data of the field notes, which are available here[1], and the confirmation of the field notes by the participants.

[1] https://klute.com/research/2024-SPL-Study/FieldNotes_Interviews_public.zip

To achieve dependability, all interviews were conducted by the same author with identical questions. The field notes were written down directly and later checked and confirmed by the participants for correctness and completeness. Within the RoboCup SPL league, the results might be transferred to other SPL teams, especially as a large proportion of the entire SPL teams have already been interviewed. It might not be easy to transfer the results to other RoboCup leagues, as these leagues have for the most part evolved independently of each other and the state of the league, rules and available tools may well be different.

5 Findings

5.1 Teams, QA responsibility and scientific background

We conducted interviews with relevant people from eight RoboCup teams (B-Human, Bembelbots, Dutch Nao, HTWK, HULKs, Naova, rUNSWift, and our team, the Nao Devils), with all teams participating in the RoboCup Standard Platform League (SPL). All teams are affiliated with universities (University of Bremen/DFKI (DE), University of Frankfurt (DE), University of Eindhoven (NL), HTWK Leipzig (DE), University of Hamburg (DE), ETS Montreal (CA), University of New South Wales (AU), and TU Dortmund (DE)). The roles of the participants were Team-Leads (4/8), Tech-Lead (2/8), Dev-Lead (1/8), and Org-Lead+Developer (1/8). Their current academic rank ranged from Student (2/8) over Bachelor (3/8) and Master (2/8) to Phd (1/8), all from the fields of Computer Science (5/8), Artificial Intelligence (1/8), Software Engineering (1/8), and Electrical Engineering (1/8). Most did not write any thesis on RoboCup (7/8) except for one master thesis. Some have written or are writing papers on RoboCup topics (3/8). The time spent being part of the RoboCup community ranged from 2-5 years (4/8), 6-10 years (1/8), 11-15 years (2/8) to more than 20 years (1/8). The majority of the teams taking part in this survey have existed since the Standard Platform League was founded or have previously participated in the predecessor league. Six teams have existed for over 14 years (6/8), only the HULKs (2013) and Naova (2017) were founded later. The team size at the time of the interviews varied between seven and 23 people (M=15.4, S=5.5). We tried to appromixate the number of contributions to scientific research based on the number of publications by the teams. This data was unknown to most of the interview participants and therefore we used publicly available information on the teams' websites to make a rough estimate. All publicly listed publications since the founding of the teams up to and including 2023 were taken into consideration. The number of publications totals around 300 for the teams interviewed here, with an average of around 3 publications per team per year (M=2.9, S=1.3). Although there might be a much better data basis, as RoboCup asks for the scientific contributions of the teams every few years, this data is not publicly accessible and was not available to us.

5.2 Code Base

The code base of the teams we interviewed is predominantly written in C++ (6/8), followed by Rust (2/8). Rust has recently evolved into a popular programming language [11] with different advantages over C++; it has become an alternative when rewriting a code base from scratch or migrating existing code [10]. This finding correlates with the age of the code bases of the teams: The two teams having 1-5-year-old code bases are written in Rust; the others (6-10 years (2/8), 11-20 years (2/8), and >20 years (2/8)) are all written in C++. Five teams wrote their code base from scratch (5/8); the other three teams used a fork or partially forked existing code (3/8). Remarkable is that the origin of the most successful code base of B-Human, which is often used as the basis for forks by newcomer teams, is over 25 years old and was based on software written to simulate electric wheelchairs [12]. The B-Human team mentioned that large parts of their code have been rewritten over the years, so that almost no parts of the original code base are left. The sizes of the code bases are also quite different and range from <10k lines of code (1/8), 10-100k (4/8), 100-150k (2/8) up to 160k (1/8).

5.3 Team success

Table 1 shows the rankings of the participating teams from 2017 to 2024. Additionally, we have included the year of team foundation, the current code base, and its year of origin.

Table 1. RoboCup placements of surveyed teams.

		Latest code base		Placement[1]					
Team	Founded	Origin	Lang.	2017	2018	2019	2022	2023	2024
B-Human	2006	1998	C++	1	2	1	1	1	1
HTWK	2009	2009	C++	2	1	2	2	2	2
rUNSWift	1999	1999	C++	5-8	5-8	3	3	3	8
HULKs	2013	2021	Rust	5-8	4	5-8	5	4	4
Nao Devils	2008	2015[2]	C++	3	5-8	4	4	5	6
Bembelbots	2009	2009	C++	13-16	9-12	5-8	7	8	7
Dutch Nao	2004	2023	Rust	9-24	9-20	9	10	C4[3]	C5
Naova	2017	2017[2]	C++	-	13	17	13	C6	C4
# of teams				24	21	20	13	17	16

[1] Years 2020 and 2021 omitted. No RoboCup was held due to COVID-19.
[2] based on a fork of the B-Human code base.
[3] Cx means a Challenge Shield placement, the 2nd SPL league.

5.4 Simulators

The 3D simulation environments most often used are WeBots (4/8) and SimRobot (3/8), the simulator contained in the B-Human code base. One team is able to use multiple simulators, in this case WeBots, LoLa and Copelia. One team currently does not even use a simulator. Some teams (3/8) have implemented separate 2D simulators that reduce the motion part of the simulation to simple 2D movements. Thus, these are often used to simulate tactical aspects of the game. When asked about the accuracy of their simulation environments compared with reality, some teams (3/7) say that their simulator is not accurate, mainly regarding the physics simulation and the applied motion control. Two teams said the environment is quite good but not perfect (2/7), one named it quite accurate, and one team could not judge on that question. When asked if teams spend efforts to close the gap between reality and the simulator, all teams denied this. One team works on a 2D simulator to better simulate the behavior. We asked the teams if their simulator environments are capable of running scripted tests that measure correct behavior and certain defined KPIs. The answers were: no (4/8), yes (1/8), partially, but not automated (1/8), or only in the 2D simulators and only for the values that can be measured in the 2D simulator (2/8).

5.5 QA measures

When asking for QA measures in place to track and improve or at least keep the quality and playing strength of a team over the years, the teams came up with various answers: The most often mentioned measures are test games, including expert judgment (6/8) and code reviews, mostly based on feature branches including merge requests (6/8). Playing test games in the lab environment or during competitions and judging the results and events identified during such test games seems to be a widespread practice among the teams. Code reviews are also a well-known QA measure in software engineering. The teams use feature branches of their version control for development of new features or for extensive rework efforts and then create merge requests that become subject to code reviews before being merged into the master branch. A CI pipeline and automated software tests was mentioned by two teams (2/8), although one of them only uses a basic smoke test as the only software test, where a simulated test game is started, and a goal has to be scored within a certain amount of time. Three teams mentioned using at least parts of the Scrum methodology, including regular and standup meetings. Other teams mentioned that they tried Scrum, but it turned out to be unsuitable, or they considered it of little help. Two teams mentioned ticket systems and boards to organize and prioritize their tickets. Two teams explicitly mentioned that they use the simulator for testing developments. Only some teams cover parts of their code using software tests. Five teams have no or less than one percent test coverage in place (5/8). One team mentioned having below 10 percent of code coverage and not running these tests automatically (1/8). Two teams seem to use software tests more extensively (2/8), with parts of their code fully covered and others partially covered.

5.6 Benchmarking and performance measurement

We asked the teams if they benchmark their software and if they measure certain performance indicators to judge on the performance of specific skills, like for example walking speed or the ability to kick a ball into the goal. Six teams (6/8) answered that they do not have such measures in place, three of these six mentioned that they at least do some manual comparison as part of the expert judgement. One team (1/8) answered, that they do some measurements on walking, but these are very basic. Mainly they are doing manual testing on the field. One team (1/8) draws some statistics from the game controller logs of tournament games.

5.7 Roadmap + KPI for SPL

We mentioned that the Humanoid League of RoboCup has proposed a Roadmap for the league to measure specific performance indicator values during the competition games. These performance indicators would be used to detect progress of the league's teams and trigger rule changes once specific key performance indicator (KPI) values are reached. Such a mechanism can be used to foster teams' progress and adapt the league's framework conditions according to the team's progress. We also mentioned this initiative and asked the participants if this would be a feasible approach for the SPL. We received positive results, such as: 'Yes, progress should actually be made measurable' and 'Has been discussed in the tech committee for three years. The very good teams want the league to progress.'. However, problems with this approach were also mentioned: 'The league has lost one third of its teams since COVID. Apart from teams from Germany, there are too few teams being founded. This is an argument against making the league harder.'

5.8 Next best steps for the teams

We were able to ask six participants for the next best steps for their teams. Two teams (2/6) stated that more resources and more developers are an urgent need. Improving different aspects of the vision system, like line recognition and sensor fusion, was mentioned three times (3/6). Improving robot skills, such as better passing and a faster goalie, was mentioned once (1/6). Removing the need for robot calibration to adapt the robots to different lighting and playground conditions was also mentioned (1/6). One team mentioned, that better processes and more testing would be important (1/6).

All in all, the answers show that improving the robot software to better adapt to different playgrounds and lighting conditions and expanding the team to include more developers are the most important topics.

5.9 Next best steps for the league

When asked about the next best steps for the SPL league, six participants mentioned several interesting ideas: The idea of a shared simulator was mentioned

two times (2/6), where teams could virtually play against each other without dealing with technical issues and robot hardware wear and tear. One participant (1/6) voted against an additional shared simulator because, from his point of view many teams are already limited by time and human resources, and thus an additional task of adapting and maintaining integration of their code bases to an additional simulator could lead to overstrain.

Two participants (2/6) mentioned that a general roadmap for the league would be beneficial. It is also mentioned by two participants (2/6) mentioned that the goal was to start measuring performance and measurability in general to identify the league's progress. Additionally, the game states 'initial' and 'ready' could be removed (1/6) as robots become more autonomous and can enter the pitch automatically, identify whistle and referee gestures to lead the game. The game controller software currently guiding the game could be removed later (1/6).

One participant mentioned improving team play (1/6). Team play between robots requires coordinated behavior, and passing the ball between robots has already been introduced as one challenge for the league. Currently, '1 or 2 teams at maximum are capable of playing controlled passes' and this should be extended. The amazement of external spectators was also mentioned as a reason why improving team play would be beneficial. One participant (1/6) mentioned that publishing the complete source code, including additional tools, could be beneficial. The teams participating in RoboCup must partially publish their source code after participation [13, p. 43]. However, parts of the source code are permitted to be omitted.

One idea was to migrate to different and better robots (1/6) as soon as better hardware at a reasonable price becomes available, to overcome the current limits of the league set by the robot hardware.

5.10 CI pipeline for automated testing

After completing the interview questions, we presented the teams with an CI-pipeline approach for automated, simulator-based testing and performance measurement and monitoring, which is described in more detail in [14]. In general, all participants found the approach to be suitable ("definitive suitable", "interesting, impressive", "for sure"). Two experts expressed concern about how the results would relate to reality ("How close is the simulation to reality?", "Useful, but questionable whether this is meaningful in reality. But, generally, a quite good criterion for detecting changes by code submissions."). Three participants expressed interest in adapting the solution ("Yes, that it has the potential to catch lots of issues.", "interesting for own team, but would need to change how time is invested in the team"). One noted that he would "partly adapt" the concept and measure and monitor code quality indicators, like test coverage.

6 Discussion

Based on our findings, we will discuss their implications for our research questions.

6.1 QA responsibility, code bases and team success

RoboCup SPL teams exhibit diverse structures, typically composed of students and researchers from academic institutions. This aligns well with the educational aspect of RoboCup but results in constant turnover of students, researchers, and even team leaders over time. Teams facing such turnover need to prioritize internal knowledge management and QA practices to maintain their playing strength. It turns out that teams do not have a dedicated QA role; instead, QA responsibilities are integrated into other roles, such as Team Lead, Tech Lead, or Developer. Due to the experimental nature of RoboCup, we did not expect explicit QA roles, but this integration means that students or researchers must manage QA alongside other, possibly higher-priority tasks. We observed that every team employs certain strategies to organize their development and teamwork, including addressing QA topics. The educational background and project tenure of team members responsible for QA are particularly noteworthy.

The majority of teams use code bases developed in C++, although newer teams are increasingly adopting Rust for its performance and safety features [11]. While five teams have created their code bases from scratch, three have built upon forks of existing projects, such as the B-Human code base. Code base sizes vary significantly, ranging from fewer than 10,000 to over 160,000 lines, reflecting the diversity of approaches and histories among teams.

We note that the two top teams in the SPL in recent years (B-Human and HTWK) have consistently maintained their playing strength and achieved high placements, securing first and second places over the last five years (volatility of 0.37). This top group is followed by three teams (rUNSWift, HULKs, and Nao Devils) that often achieve third or fourth place or at least fall within the 5th-8th rank (18 out of 18 placements across six years). However, their placement volatility is three to four times higher. For example, rUNSWift has achieved third place three times in the last four RoboCups, firmly establishing itself as a top team since 2019. These results suggest that the best teams are able to sustain or achieve appropriate playing strength during key games, enabling them to defend their placements. This consistent performance highlights the importance of exploring their QA approaches as a potential factor in their sustained success.

6.2 Approaches to QA and performance measurement

Teams rely heavily on manual approaches, including test games and expert judgment, to assess and maintain their playing strength. Expert judgment, conducted during test games, serves as a primary QA mechanism, offering qualitative insights into performance without requiring additional tools. Many teams also use code reviews based on merge requests to ensure software quality. Only two teams

reported using CI pipelines or automated tests, and these were primarily limited to basic smoke tests or individual code modules.

Although some well-known 3D simulators are used for development, they are not used for permanent quality assurance. This may be due to the lack of accuracy; motion control topics in particular are considered to be latently inaccurate. Accordingly, the gap between reality and simulation is not addressed, as this would entail additional complexity and time expenditure. Some teams have avoided this gap and developed simplified 2D simulators to simulate game behavior and tactics, which are used by individual teams to assess game tactics.

In general test coverage remains minimal for most teams. Merge requests and code reviews are common practices in software engineering and are often integrated into development workflows. Thus, we can conclude that teams actively use certain aspects of software engineering practice. However, the lack of software testing in general indicates that this is not a high-priority topic, and teams appear to accept the trade-offs associated with limited test coverage.

An example of the negative impact of inadequate automated testing and performance measurement became evident in the case of rUNSWift. The team performed well in three consecutive RoboCups (2019, 2022, and 2023), achieving third place, but their performance significantly declined in 2024. When asked about the reasons, they explained that issues noticed during a friendly match before the 2024 RoboCup led them to overreact, making major, untested changes to their source code. These changes turned out to be problematic. Although the team eventually reverted to the original version after losing four games, they concluded that 'major changes should have been tested even under time pressure.' Interestingly, even the most successful teams primarily rely on expert judgment and code reviews. This suggests that while these strategies can be effective when applied correctly, there remains significant room for improvement.

However, there is much room for improvement: Expert judgment is subjective and lacks normalization. Without normalization, it is difficult to compare assessments over time. Automated measurements of key performance indicators (KPIs) would enable teams to monitor performance trends and detect improvements or degradation systematically. Standardized performance indicators could also facilitate inter-team comparisons, fostering a more data-driven approach to QA.

In summary, the SPL league teams lack comprehensive performance measurement systems, leaving room for significant improvement. Implementing benchmarking metrics to measure aspects of playing strength is considered valuable by most participants. We propose that the SPL league define basic metrics of playing strength and provide tools to measure them. These tools should be framework-agnostic, allowing teams to adopt them without excessive effort, and should support additional team-specific metrics for customization.

Initial work in this direction has been published: After our initial publication on an exemplary CI pipeline for automated testing and performance measurement [14], the topic was taken up by the B-Human team and a first bachelor thesis [15] was written.

6.3 Structured software development lifecycle

Most teams have adapted elements of software development methodologies, such as parts of the Scrum framework. These elements, like daily standup meetings and weekly planning, are primarily used for team communication and development coordination. However, teams generally do not follow fully structured software development lifecycles (SDLCs), as are often used in industrial software development [16]. A structured SDLC involves explicit phases, such as requirements engineering, design, development, testing, and deployment, forming an iterative cycle. This approach could support the development and maintenance of complex systems like robotic software by integrating QA into every phase. However, due to the experimental nature of RoboCup, the resulting software often falls into the category of research software. Such systems are prone to problems, including high complexity, limited documentation, and insufficient emphasis on QA and maintainability.

6.4 Challenges and proposed solutions

The most relevant challenge is the robotic software for the competition itself. Regular rule changes and additional technical challenges in the SPL demand significant development resources. As a result, teams have limited capacity to focus on QA and automated testing. However, once established, QA tools could prevent bugs and performance regressions, reducing the resources needed for debugging and optimization. The educational nature of RoboCup introduces varying expertise levels among team members. Coupled with the lack of robust QA tools, this poses a constant risk of introducing bugs or performance-decreasing code.

6.5 Role of league guidelines and structure

The league's structure plays a critical role in shaping QA and performance practices. Teams expressed support for league-wide initiatives to develop common tools. Proposed solutions include shared simulators and standardized performance metrics. However, concerns were raised about overburdening smaller or newer teams, emphasizing the need to balance progress with accessibility.

7 Limitations

In this study, we interviewed only SPL teams. Approaches from other RoboCup leagues were not explored, even though they could provide additional insights. However, our focus was specifically on the landscape of SPL teams and their approaches. Additionally, we did not interview all SPL teams; instead, we selected a subset of teams based on their relevance to our study and their availability during the interview period. As a result, the findings should be interpreted with the understanding that they reflect only the perspectives and practices of this subset of teams.

The overall aim of this study was to gain insights into the approaches of successful SPL teams, and we believe this objective was successfully achieved.

8 Conclusions

The survey shows that teams currently rely on basic and mostly manual strategies to assess their quality, performance, and playing strength. Structured software quality and performance measurement practices remain uncommon and are yet to be adopted by most teams.

Most teams utilize a combination of code reviews and expert judgment by observing robots during test matches. While this is a simple and effective approach to quality assurance, it lacks the rigor and repeatability of automated and structured methods. One significant barrier to adopting automated QA practices appears to be the initial effort required to set them up and maintain them as development progresses. Another major factor is the lack of resources, coupled with low prioritization, as teams focus on features directly relevant to game performance.

RoboCup as an organization needs to adjust the league's framework and rules to align with its long-term goals, while considering the overall progress of the league. We propose that RoboCup, particularly the SPL league, define basic metrics of playing strength and provide teams with tools and methodologies to automatically measure, monitor, and compare their performance.

References

1. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: RoboCup: The Robot World Cup Initiative. 1997. Proceedings of the First International Conference on Autonomous Agents, pp. 340-347 https://doi.org/10.1145/267658.267738
2. RoboCup website, archived version of December 1998 https://web.archive.org/web/19981201174356/www.robocup.org/overview/22.html.
3. RoboCup History, https://www.robocup.org/a_brief_history_of_robocup. Last accessed 14. Oct 2024
4. RoboCup SPL History, https://spl.robocup.org/history/. Last accessed 14. Oct 2024
5. Ferrein, A., Steinbauer, G.: 20 Years of RoboCup, A Subjective Retrospection. 2016. In: KI - Künstliche Intelligenz 30, pp. 225-232 https://doi.org/10.1007/s13218-016-0449-5
6. Verner, I.M.: The Survey of RoboCup' 98: Who, How and Why. RoboCup 1998. Lecture Notes in Computer Science, pp. 109-119 (1999) https://doi.org/10.1007/3-540-48422-1_8
7. ISO 10218-1:2011: Robots and robotic devices - Safety requirements for industrial robots. International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/51330.html
8. ISO/TS 15066:2016: Robots and robotic devices - Collaborative robots. International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/62996.html

9. Lincoln, Y. S., & Guba, E. G.: Naturalistic Inquiry. 1985. Beverly Hills, CA: Sage Publications, Inc.
10. Fulton, Kelsey R., Chan, A., Votipka, D., Hicks, M., Mazurek, Michelle L.: Benefits and Drawbacks of Adopting a Secure Programming Language: Rust as a Case Study. In: Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). https://www.usenix.org/conference/soups2021/presentation/fulton
11. Bugden, W., Alahmar, A.: Rust: The Programming Language for Safety and Performance. In: 2nd International Graduate Studies Congress (IGSCONG'22). https://doi.org/10.48550/arXiv.2206.05503
12. Laue, T., Röfer, T.: SimRobot - Development and Applications. In: Workshop Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2008). http://www.informatik.uni-bremen.de/kogrob/papers/SIMPAR-Laue-Roefer-08.pdf
13. RoboCup SPL Rules, https://spl.robocup.org/wp-content/uploads/SPL-Rules-2024.pdf. Last accessed 14. Oct 2024
14. Klute, T.: Introduction of automated testing and continuous metrics & KPI monitoring in student driven projects: a use case. In: RoboCup Symposium, Eindhoven, Netherlands (2024)
15. Holsten, N.: Ein Framework für die Testautomatisierung in einer Fußballroboter-Simulation. B-Human Bacelor Thesis (2024) https://b-human.de/downloads/theses/bachelor-thesis-holsten.pdf
16. Gupta, A.: Comparative Study of Different SDLC Models, In: iJRASET International Journal for Research in Applied Science and Engineering Technology (2021) https://doi.org/10.22214/ijraset.2021.38736