Testing in Robotics Student Teams - A Case Study about Failure and Motivation

Ina Schiering[1], Arne Hitzmann[2], Reinhard Gerndt[3]

[1] i.schiering@ostfalia.de
[2] arn.hitzmann@ostfalia.de
[3] r.gerndt@ostfalia.de

Abstract. Robotics competitions are a very motivating approach for project-based learning. By the requirements of the leagues and the feedback from the competitions, students are developing ideas concerning quality assurance via testing as a self-organized team. The quality requirements they addressed encompass to a huge extent the software quality model of ISO/IEC 25010. These requirements were addressed by an adequate architecture and quality was enhanced by the introduction of software tests. The main motivation for all these measurements is based on the idea that the student team is responsible for their project in a holistic sense.

Keywords: Student projects, robotics competitions, self-organization, quality assurance, testing, agile methods, project-based learning, software quality models

1 Introduction

Computer science students are lacking to a considerable extent the motivation for testing even if it is part of the curriculum. In programming courses in our experience they do not accept to take the effort of writing tests, because they believe that their software is correct. For normal programming courses that is often addressed by agile methodologies like test-driven development [1], sometimes combined with automated testing of results [2]. Test-driven development requires that tests are written before a feature is implemented to prevent defects already during the implementation [3]. In robotics testing is even more complicated because of the complexity of the systems that consist of several components. Additionally, because of the sensors and actuators of robots that interact with the physical world, tests need also to be performed in the real world, testing in simulations is not sufficient.

The central question which is investigated in this paper is how students get the motivation for thorough testing. We investigated this question in self-organised student teams that develop systems for the RoboCup robotics competition [4]. The whole team consists of about 10 to 15 students which take part in two leagues of the RoboCup. We focus here on the work for the RoboCup

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

@work league. The students work voluntarily as a supplement to their curriculum. They stay typically between 2 and 4 years in these interdisciplinary teams which consists mainly of students of computer science and engineering.

This form of project-based learning [5] is an important element beside traditional teaching approaches. Robotics competitions offer an interesting environment for student projects where students are motivated to solve complex problems nearly on their own [6]. The lecturers role in this environment is to act as experts or advisers. The competency of these self-organised teams concerning project management was enhanced by coaching them based on agile methodologies [7]. A light-weight variant of Scrum [8] was proposed to the team where the students decided which elements fit in their team situation. This methodology was a good start for the student team to get more control over their project. Important issues that the student teams perceived were the complexity of quality assurance in robotics and the alteration of hardware. In this paper we investigate how self-organised teams tackle these questions motivated by competitions as source of motivation.

This paper is structured as follows. In Section 2 we provide a short overview of the RoboCup @work competition which is in the focus of our investigation. Afterwards, in Section 3 methodologies for software and system tests for robotics are stated. Section 4 describes the system architecture and Section 5 investigates the influences of quality requirements. In Section 6 we evaluate the approaches used by the student teams for testing based on experiences in competitions. Finally, we summarize the results and outline ideas for future work.

2 RoboCup @work Competition

The RoboCup robotic competition and symposium was initiated as a benchmark to elicit and measure advances in robotics research [4]. The RoboCup @work league is the most recent extension of the RoboCup. The tasks of the respective RoboCup @work competition are related to industrial applications, specifically having a robot to navigate and manipulate work pieces in a workshop environment. The main competitions therefore are navigation, manipulation, transportation, precision placement and interaction of two or more robots [9]. The workshop setup for the competitions, typically named 'arena', consists of navigation points and service areas. Figure 1 gives an impression of the competition setup.

For competitions robots have to navigate along given navigation points autonomously and perform manipulation and transportation tasks at, respectively between service areas. Many different aspects of production situations are considered. For example, dynamic changes are considered by introduction of a conveyor belt. Robots have to master the challenges fully autonomously. However, real-world set-ups are subject to sensor noise and wheel slip and thus cause many different situations within repeated runs of the same tasks. This has many implications on the development and test of software for such systems.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

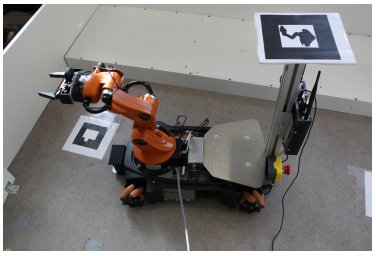

Fig. 1. Kuka Youbot robot for Robocup @Work

3 Software Testing for Quality Assurance in Robotics

For software testing of robotics we consider here techniques of the area of software testing [10] that are evaluated and adapted to the specific needs of systems that interact via sensors and actuators with the environment. The basic step to ensure quality is *static analysis* of code by automated checking tools. There syntax and coding styles can be checked to enhance the readability of the code and hints concerning potential defects, e.g. data flow anomalies, dead code, are given. The next step in software testing is typically the use of *white-box testing* to test the internal structure of software components. Because of the strong interaction with the physical world this is only applicable in few situations. Afterwards *black-box testing* is employed to test the functionality of the system based on the defined requirements.

A common approach to test such complex systems like the software structure of robots is using *simulation* for black-box testing [11]. In this environment the robot can show all its intended behaviour and failures can be observed. In robotics often *grey-box testing* is used which incorporates aspects of white-box and black-box testing. It tests part of the functionality of the system with the internal structure of the components in mind and allows to test the integration of components. It is often applied to situations where mere white-box testing is not reasonable, because the complexity of the system lies in the interaction of components and testing of isolated components is time-consuming. Based on the general idea of grey-box testing there exists approaches for random generation of test suites by Barret et al. [12].

Black-box and grey-box testing can be used as a basis for regression testing [13]. These ideas allow further on to use concepts as test-driven development where tests are defined at the same time or even before the development of the software [14]. This is typically combined with the technique of *continuous integration* of agile software development [3], where the software is built and static analysis and tests are performed several times a day, e.g. each time a developer checks in the software in the revision control system. Concerning quality assurance of software in robotics in general Koo Chung et al. [15] propose an approach for quality assurance in robotics based on ISO 9126 (now replaced by ISO/IEC 25010 [17]).

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

4 System Architecture Based on ROS

The Robot Operating System (ROS) framework [16] was chosen as the overall software structure for the RoboCup @work robot. The ROS framework is based on a blackboard architecture. This blackboard is maintained by the central program of every ROS system, the *ROS-Core*. Every sensor is publishing its data to topics on the ROS-Core where other programs, which are called nodes in the following, can subscribe to these topics and receive the data they need.

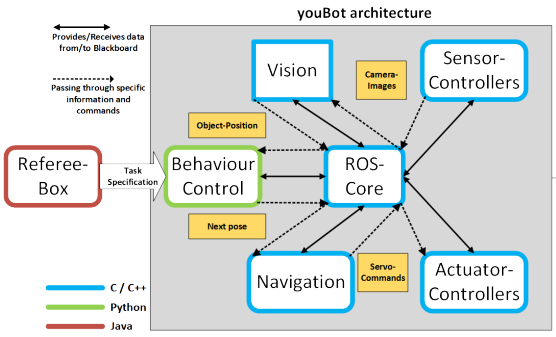

Fig. 2. Software architecture of the robot

The whole ROS system is network based so distributing nodes to different machines is an easy task as long as the network connection between the nodes and the ROS master remains stable and sufficiently fast.

5 Quality Requirements Influencing the Architecture

Based on the ideas of Koo Chung et al. [15] to use ISO 9126 as the basis for a quality model in robotics, we employ the "software product quality model" of the subsequent standard ISO/IEC 25010 [17] and investigate which of the quality characteristics the team addressed in the software architecture. It is an interesting observation that there are examples for measurements for most of the characteristics. The student team developed this architecture on their own initiative motivated by the rules of the competition [9] and problems they perceived in competitions.

- *Functional suitability:* To address the problem that during travel the sensor mounts are often bent during transport, the student team print sensor mounts with a 3D printer for accuracy. Hence afterwards they can assure that the sensor positions are accurate.
- *Reliability:* To enhance reliability especially concerning the subcharacteristic fault tolerance and recoverability failure situations are monitored and recovery behaviour is introduced.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

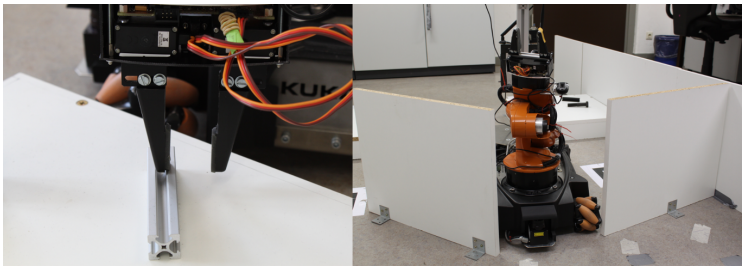

Fig. 3. Actually occurred misbehaviours (f. left, missing the object while grasping it and r. crashing into arena)

- *Performance Efficiency:* Since the computational power of the robot is limited and due to the time limit for each challenge, resource utilization and timing are important design goals for the nodes.
- *Operability:* (not addressed)
- *Security:* (not addressed)
- *Compatibility:* The basic aim of the blackboard architecture used is modularity.
- *Maintainability:* The robot allows to exchange parts like sensors which are subject to change very often and the software assumes that some kind of sensor is to be found under a designated port.
- *Transferability:* The use of configuration files for e.g. the description of service areas, delays for camera stabilization, supports the easy parametrization of the behaviour for different arenas. The ROS software stack used by all teams ensures that different teams can exchange software. There is an idea to mount LEDs around the camera to render the vision less dependant of the current light situation.

Concerning the criteria of the system quality in "use model" of ISO/IEC 25010 the system addresses the attribute of *safety*. According to the rules for the competition, the robot needs to have an emergency stop, also the robot is not allowed to leave the arena.

6 Increasing Quality by Testing

Year 0 - Competition: The student team received the robot and started the development of the software based on the ROS core three weeks before they planned to participate in their first tournament in the late spring of 2012. The main goal during this time was to produce a working behaviour and a code base that allows the robot to solve the tasks navigation and manipulation. During this time no quality assurance took place in the development process. Most of the code was written by one developer in Python.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

Year 1 - Evaluating Testing Methodologies: When the team returned from the competition it turned out that nobody was familiar with the code any more. Additionally, the code was difficult to adapt, because hard-coded values where spread all over the code. The team addressed this by starting from scratch,

Methodology	**Remarks, Tools**	**Result**
Static Analysis	cpplint, (valgrind)	useful
Continuous Integration	Jenkins (bugtracker, revision control system)	useful
White-Box Testing	used for the Vision	partly useful
Grey-Box Testing	Simulation	partly useful
Test-Driven Development	AR-marker, tracking	useful

Table 1. Testing methodologies of student team

introducing configuration files into the architecture instead of hard-coded values as described in Section 5. Also, they introduced static analysis to ensure coding styles and get feedback about code with potential defects. All the nodes written in C or C++ are tested by a continuous integration server using cpplint for static analysis. There also valgrind, a tool to detect potential memory leaks, was introduced. For documentation of work the team introduced beside the continuous integration system also a bug tracker and a revision control system for the software.

Another critical aspect concerning quality assurance is the use of Python for many central nodes. With Python being an interpreted language, without the checks of the tool chain of compiler languages, failures will only become obvious by means of an exception, if the respective code is executed. This problem has been partially addressed by means of parsing the configuration files with an external program after every change.

Most of the students have a strong background in software engineering. When agile methodologies were introduced, they started to evaluate software testing techniques in the field of robotics. Hence they perceived that white-box testing is only applicable in few situations, since the interaction of components is in the focus. An example of white-box testing used by the student team is the vision, where pre-recorded pictures with annotations can be used for isolated tests. Additionally, a simulation was used as a way to test the behaviour of the whole system as a form of grey-box resp. black box testing.

Year 2 - New Ideas: It was witnessed that even the lightest changes of the environment led to different path chosen by the navigation and changed the way of grasping objects in the physical world. With simulation it was not possible to reproduce this behaviour. So an approach was chosen that uses the real robot for repeatable testing (see Laval et al. [14]). The student team developed a system based on AR-markers and off the shelf cameras to track the robot in the arena.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

Fig. 4. AR-marker tracking setup

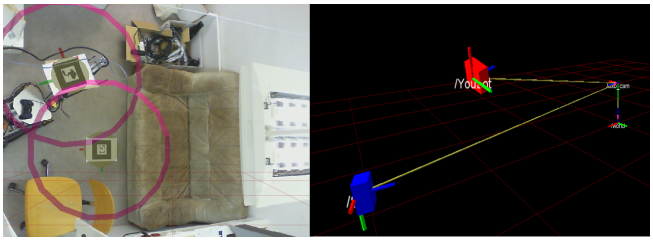

Fig. 5. Screenshot of tracking visualisation

The fact that markers are used in the @Work league led to the approach of using AR-markers attached to the robot to track it and identify the service areas the robot has to drive to. With AR-marker it is possible to reconstruct the relative rotation and translation of a marker to the camera by one camera at a time. To build up the test environment special AR-markers were prepared for the robot and its destinations. Then a setup of cameras were installed to be able to observe the whole arena (Figure 4). With the help of this setup test-driven development could be introduced.

7 Conclusion and Future Work

In the student team the motivation for thorough quality assurance and testing grew over several steps fostered by failure in competitions, starting competency in the field and a growing insight in the structure of defects and failures. This growing competency concerning software and system quality even influenced the way the students refined the architecture and evaluated technologies. Hence their perceptions and decisions are influenced by their experiences in quality assurance. Additionally, they even engineered a very innovative low-cost approach for black-box testing in a robotics environment. Based on these results, we will investigate the possibilities of transfer to programming courses.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42

References

1. Stephen H. Edwards. Improving student performance by evaluating how well students test their own programs. *J. Educ. Resour. Comput.*, 3(3), September 2003.
2. Kevin Buffardi and Stephen H. Edwards. A formative study of influences on student testing behaviors. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 597–602, New York, NY, USA, 2014. ACM.
3. Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004.
4. Robocup. `http://www.robocup.org/`.
5. John Dewey. *Experience and Education*. Touchstone, 1938.
6. Reinhard Gerndt and Jens Lüssem. Mixed-reality robotics - a coherent teaching framework. In Roland Stelzer and Karim Jafarmadar, editors, *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, pages 193–200. INNOC - Austrian Society for Innovative Computer Sciences, 2011.
7. R. Gerndt, I. Schiering, and J. Lüssem. Elements of scrum in a students robotics project: a case study. *Journal of Automation Mobile Robotics and Intelligent Systems*, 8, 2014.
8. Ken Schwaber. Scrum development process. In *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*, pages 117–134, 1995.
9. G. Kraetzschmar, W. Nowak, N. Hochgeschwender, R. Bischoff, D. Kaczor, and F. Hegger. Robocup@work rulebook. http://www.robocupatwork.org/download/rulebook-2013-06-08.pdf.
10. Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.
11. Son Jung-Rye, Kuc Tae-Yong, Park Jong-Loo, and Kim Hong-Seak. Simulation based functional and performance evaluation of robot components and modules. *International Conference Information Science and Applications (ICISA), 1-7, IEEE*, 2011.
12. A. Barrett and D. Dvorak. A combinatorial test suite generator for gray-box testing. In *Space Mission Challenges for Information Technology, 2009. SMC-IT 2009. Third IEEE International Conference on*, pages 387–393, July 2009.
13. G Biggs. Applying regression testing to software for robot hardware interaction. *International Conference on Robotics and Automation(ICRA)*, pages 4621–4626, 2010.
14. J. Laval, L. Fabresse, and N. Bouraqadi. A methodology for testing mobile autonomous robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1842–1847, Nov 2013.
15. Yun Koo Chung and Sun-Myung Hwang. Software testing for intelligent robots. *International Conference on Control, Automation and Systems*, pages 2344–2349, 2007.
16. Robot operating system. `http://www.ros.org/`.
17. ISO/IEC 25010:2011, systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization*, 2011.

Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics &
5th International Conference Robotics in Education
Padova (Italy) July 18, 2014
ISBN 978-88-95872-06-3
pp. 35-42