Elements of Scrum in a Students Robotics Project - A Case Study

Submitted: 5th May 2013; accepted: 30th June 2013

Reinhard Gerndt, Ina Schiering, Jens Lüssem

DOI: 10.14313/JAMRIS_1-2014/5

Abstract:

Robotics competitions allow self-organised learning in a quite natural way. In the last years, we have observed an increasing complexity in these competitions. At the same time, the search for an adequate project organisation became more and more important. As traditional project organisation methods failed, we adopted Scrum and tried to adapt this agile methodology to student projects.

Keywords: *student projects, robotics competitions, project management, self-organisation, agile methods, Scrum, project-based learning*

1. Introduction

Student projects are an integral part within our robotics teaching activities. In our teaching philosophy [16], we combine traditional learning approaches with project-based learning [9]. Competitions offer an interesting environment for student projects [12].

Furthermore, robotic competitions provide an excellent motivation for students to study in a self-organised manner, which opens widely the path for natural curiosity. Competitions provide clear functional objectives and measures of success. Consequently, requirements (in form of rules for the respective competition) and deadlines (e.g. the day when the competition takes place) are not questioned. Furthermore, competitions offer a means of assessment outside of the university grading system.

A couple of years ago, when our student groups started with participating in competitions, they were nearly self-organised. Lecturers were in the role of experts in robotics.

Since then, we have seen an increasing number of competitions with more and more sophisticated technical and scientific objectives. Reaching a sufficient quality level and good rankings in the competitions with student groups became an increasingly challenging undertaking.

Instead of then taking the role of a project manager and lead the student group, the authors chose a different approach. It was perceived that the self-organised student groups were such a success story that self-organisation should not be given up too quickly. Especially young students benefit enormously from these experiences - such as building teams, or managing changes.

In the past years, it thus became more and more crucial to find a self-organising project management approach that preserves the motivational aspects and leads to at least satisfying results in the competitions.

To address these challenges, we investigated to which extent agile methods like Scrum can be used for the management of student projects.

The remainder of the paper is organized as follows. In Section 2, we introduce the robotics competitions, our student teams have participated in. Section 3 describes the project management methodologies we applied. Sections 4 and 5 focus on agile methods and their applicability in student projects. In Section 6 we report our first experiences in using Scrum for student robotics projects. Finally, we summarise the main findings and describe future work in Section 7.

2. Robotics competitions and their complexity

Robotics competitions differ in many ways. There have been and still are competitions related to robotic cars, aerial vehicles, military robotics, just to mention some. One of the most prominent robotics competitions is the RoboCup [3]. It is based on the challenge to play a game of soccer with a humanoid robot team against the human world soccer champions in the year 2050. Many of the aspects of this paramount objective are targeted in individual leagues, some of those further subdivided into sub-leagues and partial competitions. To foster exchange with other robotic fields, some peripheral leagues and competitions, which are not immediately related to robotic soccer have also been introduced to the RoboCup. Aside of the specific functional objectives, different aspects of complexity [8] relate to the partial competitions. As the most obvious aspects, in this section, we present the targeted competitions with respect to the robotic complexity and the task complexity. The robotic complexity covers hardware and immediate, i.e. low-level control or kinematic complexity. The task complexity describes the complexity of the functionality a robot may have to implement for a competition.

2.1. Mixed-Reality competition

Initially, the student group joined the RoboCup Mixed-Reality competition [11]. The challenge is a robotic soccer game with up to 11 small wheeled robots per robot team, playing with a simulated, virtual ball. The mixture of real robots and virtual environment and ball led to the name. The main task is implementing a cooperative, possibly swarm-like, behaviour of a group of robots. Small, differential drive cubic robots with a volume of approximately 8 cubic centimetres are used as players. The playing field is

a horizontally mounted screen to display the virtual field and the ball. The robots are controlled via an optical link by software agents, running on a standard PC. Process information, like the robot position is made available to the agents by means of additional software packages.

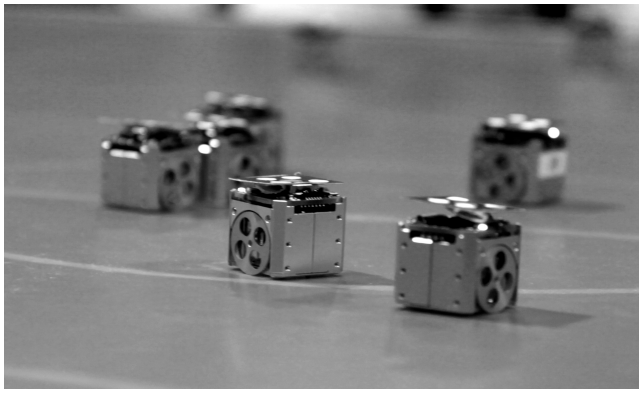

Fig. 1. Standard mixed reality robots used in RoboCup competition

The basic system was developed beforehand by a specialized development team and made available for the implementation of robotic soccer functionality by the student groups. Differential drive robots have a straight-forward kinematic model, such that robotic complexity is comparably low. The task for the student group is to implement the cooperative behaviour of a group of robots to play a game of robotic soccer in software. The limited complexity of the agents and loose coupling of system components allows for individual students implementing the entire functionality or behaviour of a robot. Thus the student group is facing a relatively low complexity at the robot and the task level.

2.2. RoboCup kid size humanoid competition

Following the initial successes, the significantly more complex RoboCup kid size humanoid competition [4] has been addressed as next major step. The challenge currently is a "three vs. three" robotic soccer game. Initially, the main task is realising robots that are capable of kicking a ball to a goal in a more or less sophisticated way. The size of the humanoid robots with 18 or more drives is in the range of 30–60 cm. The field is six times four meters. A tennis ball is used to play the game. Goals, the ball and the robot teams are colour coded. The robot is controlled by one or more on-board computers and carries all its human-like sensors.

Designing and building the robots now is part of the competition. In addition, low-level control and kinematics became significantly more complex. Furthermore, acquiring information on the environment in sight of unreliable data from the sensors and wear of hardware adds to the robotic complexity. The task complexity is basically comparably to the complexity in the Mixed-Reality competition. However, introducing a real ball slightly added to the task complexity.

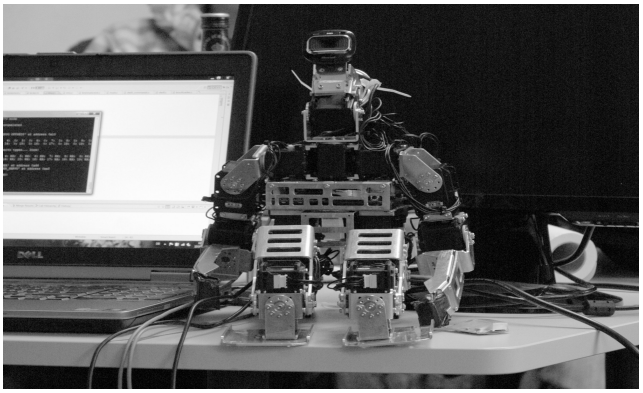

Fig. 2. Current humanoid robot, adapted from open source robot plans

As a general property, the humanoid robots require a closer cooperation at software and hardware level and at the hardware-software interface and thus required closer cooperation between members of the development team.

2.3. RoboCup@work competition

As a currently final step, the RoboCup@work competition [5] related to an industrial workshop situation with the Youbot, a miniature version of a real mobile industrial robot, has been addressed. The challenges within the competition include navigating in a workshop situation and manipulating and transporting work pieces. The robot consists of a 60 x 40 cm omnidirectional mobile base with very user-friendly kinematics and a 60 cm industrial robot arm with five degrees of freedom. It has been delivered operational with a Robot Operating System (ROS) basic software [6].

Fig. 3. Kuka YouBot robot without team-specific enhancements

Thus a basic operational software and hardware platform was available from the very beginning, like in the Mixed Reality competition. However, all sensors, like cameras and laser range scanner had to be selected, integrated and maintained by the student group. Typically software libraries were available to access the sensors. Thus the robotic complexity was lower than in the humanoid competition, but significantly higher than in the Mixed Reality challenge.

Functional requirements include localizing, identifying and grabbing different work pieces and carrying out different tasks, thus resulting a higher task complexity. With a larger community, working with the Youbot, reliability issues are less dominant. However, the large set of (sub-) functions that make up a robot task requires close cooperation of the student group at software level.

The robot and task complexities of the three competitions we participated in are summarized in the following figure.

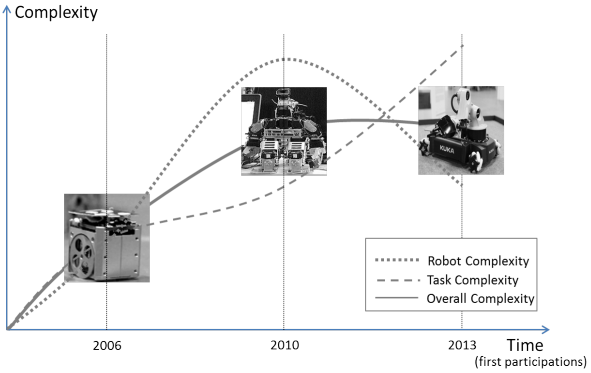

Fig. 4. Complexity of robotics competitions

3. Philosophy and the management of complexity

3.1. Teaching philosophy: roles and requirements

In robotics education we combine "traditional" learning methods with learning-by-teaching and problem-based learning approaches. For us, problem-based learning is not only an additional teaching method, it's rather the most important brick in our teaching strategy [12] which follows the European Qualification Framework.

Robotics competitions offer a wide range of problems student groups can tackle. In our teaching philosophy, students must have the chance to solve this kind of problems (practically) on their own. Therefore, we avoid an involvement in the day-to-day project work. Consequently, we act as experts and are often in the role of an advisor or mentor. So, self-organisation is our main requirement for the student group.

Further sources for requirements are the students themselves. Our more technical oriented student team aims to focus on robotics (i.e. hardware and software development).

Competition organisers are a last important source of requirements. The organisers set the rules for the competition including constraints on the robots and the underlying infrastructure.

The most important requirements are shown in Figure 5.

3.2. Managing complexity: project organisation

During the last seven years our student team participated in three different competitions (see Figure 4). At the beginning we did not pay too much attention on project organisation or project management.

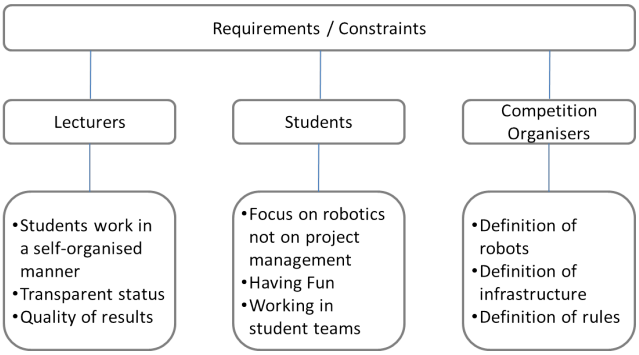

Fig. 5. Requirements and constraints

This attitude changed drastically as we started in the Kid Size Humanoid League Competition.

Mixed-Reality Competition Initially, a self-organising merely unstructured approach was chosen. Every member of the group felt responsible for the overall outcome. The specific implementation that was to be used for an official competition event was chosen by internal competition. Different competencies among group members had some influence on the selection process, but rarely influenced the decision during this phase of project management. If intermediate results, especially during an official competition, indicated the necessity for changes in the software, specific branches of the implementation have been developed from the selected version and selected by immediate comparison. In some rare cases, the group switched to one of the previously discarded implementation and carried on with it after unanimous decision.

Eventually, a hierarchical project management structure evolved. Students had to take over organisational duties, like interacting with the competition organisers and organising travel and accommodation and thus turned into acting management students. They, however, often could not cope with the high, also emotional stress during competition events sooner or later and resigned. It is worth to mention, that members with high technical and scientific competencies always concentrated on the technical and scientific work and did not take over management duties.

In the technical domain, clearly bounded responsibilities evolved and all individual tasks were covered by individual members of the team. The boundaries, however, due to the considerably low complexity and loose coupling evolved naturally and required no specific agreements among group members. Definition of interfaces was obvious and none to very little project management activities were required.

Kid Size Humanoid League Competition Driven by the success, subsequently activities in the humanoid sector have been started. Initially the same project management and iterative individualized design procedures have been used. Members of the group concentrated on their specific segment. They defined indi-

vidual measures of success and evaluated the development outcome against their own partial test cases. Often, the official competitions were the only integration tests. However, with a high degree of independence students individually prefer to add new features instead of working for quality. By constantly mixing debugging and developing new features, they jointly never reach a release that could be used as a fall-back position.

From the accomplishments it became obvious that, with the significantly more complex functionality and higher interdependencies among system components, now a more extensive project management was a necessity. Furthermore, the considerably small team relied on finding synergies to handle the overall complexity of the robotic system.

RoboCup@Work competition The activities related to the industrial robot started from a similar point. However, with a clearly defined hardware and software architecture and some basic software functionality available, the well-known iterative approach could be followed for some time. However, eventually, by improving existing hardware and software components and adding new functionality, the overall complexity rose in such a way that project management now became necessary. As a consequence an agile approach was proposed to the student group.

3.3. Student feedback

In order to confirm our impression, we carried out a survey among students with at least 6 months of involvement in the robotic work group. In one section of our questionnaire, we asked students for their priorities in the robotic projects. The results showed a clear priority for a self-organised approach over guidance by a lecturer. Priorities for individual work and teamwork were almost leveled, with a small bias towards teamwork. Spending time for meetings or individual work was leveled, like having fun versus achievements in competitions. As an interesting result students claimed to prioritise quality over adding new features, which was not fully in line with the impression of the authors, while guiding the team.

Another part of our questionnaire was dedicated to project management methods. According to the feedback, all students were quite familiar with the waterfall model. The waterfall model is an early model for software engineering where the phases requirements, design, implementation, test and maintenance are performed sequentially. V-model and the iterative approaches like the spiral model were known less and the spread of familiarity was larger. The V-model is a an enhancement of the waterfall model. In the V-model, the development of tests for the test phase is already started after the requirements are specified. The spiral model was developed based on the experiences made with the waterfall model. The central idea is that it is challenging to develop complex projects in one step. Hence the project is divided into so called interations. In each iteration a development cycle as in the sequential models is performed. Hence experiences with first prototypes and changes can be included into the next iteration.

Agile methods, e.g. Scrum were known even less with a considerable large spread of familarity, even after some exposure during the project. However, voting on the expected suitability of the respective methods showed a clear preference for agile and a little less for iterative models. V- and waterfall models were considered not suitable. In general the findings correlate with the project management approaches that have been used so far and currently are used (fig. 6).

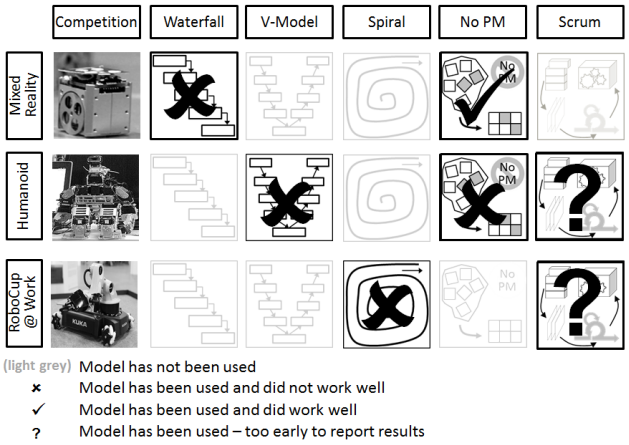

Fig. 6. Project management approaches in the students' team

Agile approaches and their usage in our student group will be discussed in detail in the following sections.

4. Agile Methodologies

The investigation of agile methodologies was started as a reaction to experiences made with the waterfall model in the 1990s based on ideas of lean production. At around the same time the imperative programming paradigm was accompanied and in parts replaced by object oriented programming. Also the phase of the so-called new economy started which led to shorter time-to-market and therefore also to shorter product life cycles and frequent changes of requirements during projects. These requirements were difficult to realise with existing methodologies like the waterfall model, but also with the iterative methodologies.

The aim of agile processes developed in this period was to create an environment that facilitated the collaboration of people and tried to enhance the productivity of developers. The focus is on creating value for users and allow for recent changes of user requirements. The most known agile methodologies developed by this time were XP [14] (eXtreme Programming) and Scrum [19]. For an overview of agile methodologies and a review of studies about adoption and experiences of these methodologies see [10]. XP and Scrum are both often used in teaching environments, which are specialised courses concerning XP

[15], general software engineering courses [20] or the management of student projects integrated in courses [18], [17]. Since these case studies were very positive in general, we tried to adapt agile methodologies to self-organised student groups.

The agile manifesto describes the values incorporated in agile methodologies [7]. The following values are the basis for agile methodologies and processes:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

In the following we give a short introduction to XP and Scrum which are the best known examples of agile methodologies.

4.1. XP

eXtreme Programming (XP) was developed by Kent Beck, Ward Cunningham et al. [14]. The focus of XP is on the close communication with users, extremely short development cycles which are called iterations. They are typically 1 to 2 weeks long. During this iteration the typical phases of software engineering (analysis, design, coding and testing) are performed. We present XP here by stating the XP practices according to Beck [14] which are often used also outside of XP:

- *The Planning Game* — During the planning game the next release and the tasks of the iteration are planned.
- *Small releases* — Small releases are realised that are put into production quickly. It is possible to have e.g. daily releases or weakly releases.
- *Metaphor* — The system as a whole and all components should have names that are easy to remember and relate to the use of the component.
- *Simple design* — The architecture of the software should be as simple as possible. There are no preparations for future features. Additional complexity that is no longer needed is reduced by Refactoring.
- *Testing* — Programmers write unit tests to test their code continuously during development. These tests must run flawlessly to demonstrate that a feature is finished.
- *Refactoring* — Programmers restructure the code and adapt the internal structure of the system as often as appropriate without changing its external behaviour. This can be assured by testing. The combination of Testing and Refactoring is called Test Driven Development.
- *Pair programming* — Two programmers work together. One programmer is writing code while the other one reviews it, thinks about improvements and is able to give immediate feedback. The roles are switched regularly.
- *Collective ownership* — The code is collectively owned by the team. Every programmer has access to the whole code and is able to change code everywhere.
- *Continuous integration* — The software is build and integrated regularly. This could be done several times a day or at least always when a developer completes a task.
- *40 hour week* — To work overtime should be avoided as a general rule, because the quality of the code would not be appropriate when written in a stress situation.
- *On-site customer* — It is expected that real users are available the whole day for the team to answer questions.
- *Coding standards* — Write Code according to agreed Coding Standards. The resulting code should be easy to read.

4.2. Scrum

Scrum was developed at the same time as XP (amongst others) by Jeff Sutherland and Ken Schwaber. This software development framework consists of the following elements (see [13]): Scrum is based on short releases of about 2-4 weeks which are called *Sprint*. To organise the development in these short releases the following elements are defined:

- **Roles:** Product Owner, Scrum Master, Team
- **Ceremonies:** Sprint Planning, Sprint Review, Daily Scum Meeting
- **Artifacts:** Product Backlog, Sprint Backlog, and Burndown Chart

The *Product Owner* is the representative of the users. The responsibility of this role is to define the features to be realised. These features are described and prioritized in the *Product Backlog*. This is often realised in the form of User Stories. Beside the prioritization the complexity of the user stories is estimated. This is often realised by estimating not the effort of user stories but the relative complexity. To avoid the influence of other participants often a so called planning poker is used. At the end of the Sprint the Product Owner has to check and accept the results of the Sprint.

The *Scrum Master* has to ensure that the collaboration inside the team leads to cooperative work and that the Scrum process is followed.

The team organises the development work and defines the *Sprint Goal*. This is based on features of the Product Backlog. The selected features are transferred to the *Sprint Backlog*. The team plans the development work in the *Sprint Planning* at the beginning of the Sprint. There, concepts concerning the architecture and ideas for tests are detailed. Also open issues can be clarified with the Product Owner.

Based on this plan the team is responsible to finish the Sprint Goal at the end of the Sprint and to present the results to the Product Owner. The progress of the team is denoted in a *Burndown Chart* where finished

user stories are marked. A user story is finished, if it is "done". This notion of done incorporates thorough testing of the solution. Every day there is a short (stand-up) meeting, the so called *Daily Scrum* to discuss questions, talk about the progress and organise the work of the day. In Scrum the team chooses appropriate methods for the development work. There are no regulations concerning the organisation of software development from the Scrum methodology. Often some of the practices of XP are used, but these practices are selected by the team and are not mandatory.

At the end of the Sprint there is a review of the results and the process in the *Sprint Review*. Afterwards the next Sprint starts again with the Sprint Planning.

5. Elements of Agile Methodologies Proposed to Student Group

The specific situation in a mostly self-organised student project imposes a number of constraints on the selection of the project management elements. However, the necessity of a project management to handle the technical complexity and organise the work sets limits to the constraints. As a result a not fully homogeneous set of elements of agile methodologies has been identified as a base for project management in such projects.

Fig. 7. Scrum Meeting of Robotics Team

From the experience of project management in student projects a very critical point was the difficulty in estimating efforts. The reasons are the lack of experience and that the projects are very innovative and hence the technical challenges are not clear at the beginning. Furthermore, hardware failures and the resulting procurement of replacement parts often introduce delays. Therefore, project plans were typically too ambitious and the students left work packages semi-finished, because of the pressure of the plan.

Another important area are the special aspects of working with students: The work must be fun, because the students are working voluntarily. Also they are pursuing their studies and are attending lectures and doing study projects as a main priority. Hence the time they got for the robotics activities is difficult to plan and interfaces between work packages are critical. Therefore team building and flexible planning are important aspects.

An important point is the communication not only inside the single activities but the transparency and sharing of knowledge between the activities. The idea here is to try to build upon common experience e.g. concerning architecture like the Robot Operation System (ROS), blackboard architecture and artificial intelligence.

Therefore a project management methodology for self-organised teams of students in robotics projects should address the following aspects:

- Easy estimation of work packages
- Innovation oriented flexible project planning
- Quality checks and testing
- Team building
- Transparency

In general the values of the agile manifesto stated above are well suited to address the special challenges of student projects as stated in the description of the requirements: The individual and the team are in the focus of the work, recent changes to the plan are accepted and working software is the aim. Also quality management is incorporated and the agile methodologies are inherently transparent. Therefore we proposed to the student group to use an agile methodology for project management as stated above.

For the decision concerning the agile methodology to use we considered the following aspects: XP is very much concentrated on organising the software development itself by prescribing engineering practices, whereas Scrum allows the team to choose the practices for development themselves. Also XP assumes that a user is always available (On-Site Customer) whereas Scrum accepts that a representative is available at least during Sprint Planning and Review. Since some of the XP practices as Pair Programming, On-Site Customer and the strict use of unit tests for Testing are difficult to realise in a robotics project, we decided to use a variant of Scrum adjusted to the special requirements of self-organised student projects.

We started with Sprints of 2 weeks length and user stories to formulate a Product Backlog. With this form of a rough planning in general and a detailed plan made just at the beginning of the sprint it was easier for the students to estimate efforts and plan communication and interfaces in the project. The project planning in the form of Sprint Planning turned out to be more realistic because the students had to work on their User Stories (work packages) until they were finished and had to explain progress and difficulties to their fellow group members where fruitful discussions followed. But we observed that until now often User Stories were not finished in one Sprint which is an issue.

Daily Scrum Meetings are not possible, because the students do not work on a daily basis on their projects. The compromise was the relatively short length of the Sprints. We used at least weekly meetings for the last weeks before the competition.

The testing and quality checks which are the basis for the definition of done of a user story were difficult to realise: We used continuous integration and coding standard as elements of XP for improved code quality. But the use of test driven development and automated unit testing in general is not possible to this extent in a robotics environment. Hence the quality checks were reduced to simulations where possible and tests with the robots, which is quite resource consuming. Also we perceived the issue that mechanical elements broke or at least changed their behaviour over time, e.g. due to wear, which makes the notion of done difficult.

The role of the Product Owner could not be realised until now. Hence the only external checks are the competitions and some workshops. This is not sufficient and we are evaluating other ideas as getting externals as Product Owner, having group events where results are presented or planning presentations to international guests of the university etc. as a substitute. At the moment the Sprint Review is realised in form of a group meeting accompanied by the advisers.

The students experienced the Team role as very helpful. The role of the Scrum Master was taken by one of the authors in the form of a coach.

Instead of Burndown Charts and Scrum Boards we used the ticket tools Redmine [2] in combination with the continuous integration system Jenkins [1] as a source of continuous feedback to the group. Since the parts of the group working together are relatively small it was important to realise the transparency of all projects. The Product Backlog and the Sprint Backlog were also realised via this ticket tool. A monitor with an actual status of the projects is placed in each laboratory. The central idea was to choose tools that are open source and used frequently for the organisation of software projects.

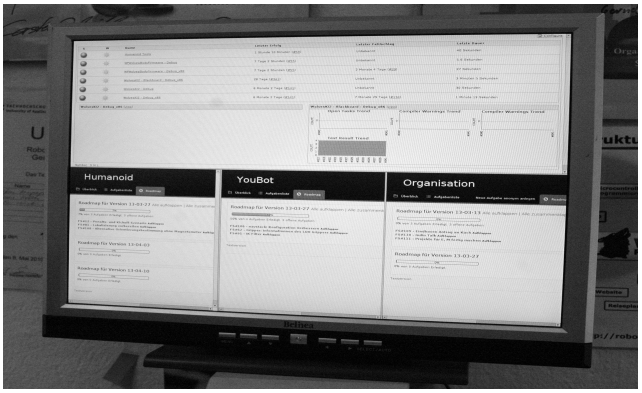

Fig. 8. Status of Projects in the Laboratory

6. Experiences and Ideas for Adjustments

In general, the feedback of the students was very positive. They appreciated the transparency and communication. The regular meeting was excellent to create a team spirit and allowed them as a group to lead the project. Hence it is a very promising approach for self-organised student projects in robotics. In the following we discuss our experiences and present the ongoing discussion with the group about adjustments of the methodology.

With a group of volunteers with changing time budgets, the stringency of the Scrum philosophy is problematic. Students are not available full time and sprints may need to be re-adjusted dynamically to account for unforeseen external and internal to the project distractions. External distractions may result from specific requirements of students to follow lectures, prepare assignments or earn money for a living. This specifically holds for undergraduate and graduate students. As a major internal source of distraction, we identified hardware failures, which required lengthy repair or ordering of replacement parts. Based on the observations, any unforeseen event to cause an extra work load of about 2 days is considered as a distraction. Apparently, any individual shorter distraction can be mostly compensated within a typical sprint period. However, if distractions result in delays and not meeting the time-line too often, the self-improving estimation of effort for specific tasks is jeopardised. Furthermore, not meeting deadlines becomes a regular case. Having a set of identical hardware unit available helps to carry on with a task with less delay. However, identifying deviations from an expected behaviour as a hardware issue still consumes time. Furthermore, hardware units need to be in an equal state. This, however, may be a challenging if not infeasible requirement for hardware systems that suffer from wear or from performance deterioration due to consumption of material, building up of heat or other physical effects.

Fig. 9. RoboCup team during a competition

On the other hand, agility very much is in line with the expectations of a group of volunteers working on a project. Agile methods account for a high degree of self-organisation. With their typical culture of frequent meetings, they foster a high degree of transparency. However, lacking a clear *Product Owner* role during the development phase negatively affects the quality of the results. Often only the competitions take over a *Product Owner* role to identify the major shortcomings of the current implementation. For many of the RoboCup student groups, the authors observed the most agile development phase between the national competitions in spring that communicate clear user

requirement and the international event in summer. Additional smaller events may contribute to a 'virtual' *Product Owner*, but need to be considered carefully with respect to the objectives and the time budget.

One of the major problems before the introduction of Scrum was project planning and the estimation of work packages. We noticed that User Stories and discussing only the next Sprint supported the students in the group to structure their work better than before. At the moment many of the User Stories are too long to be finished in one Sprint (so called *Epics*) and therefore could not be finished in one Sprint as intended. This was a compromise since the group realised a complete change to a blackboard architecture. Concerning this issue we will need to extend the length of Sprints from 2 to perhaps 4 weeks and to get experience how to break down User Stories. This is one of our main goals for the next period. Additionally, the group has to decide how often they want to meet to discuss their actual tasks for status and feedback. Since the student group has now experiences in the methodology they are able to decide as a group how to adjust Sprints and additional meetings.

In the third section of our questionnaire, students were asked for recommendations to organise the Scrum approach. There was a clear vote for sprints of two weeks with 'Daily' Scrum meetings once a week. This is in line with an average involvement of about 2 days per week in the robotics group activities. Hence the student group needs to investigate how to break up user stories and must concentrate on estimation of user stories. As an additional tool the Planning Poker will be evaluated.

Although students were often not able to finish their task during the Sprint, the regular meetings helped them to explain the reasons to the entire group and discuss the status openly and straightforward because of the notion of 'done'. Hence the regular meetings improved the communication in the group and the transparency between the different activities. Some of the students addressed that they would like to try a Scrum Board instead of the solution with Redmine used at the moment to enhance transparency further.

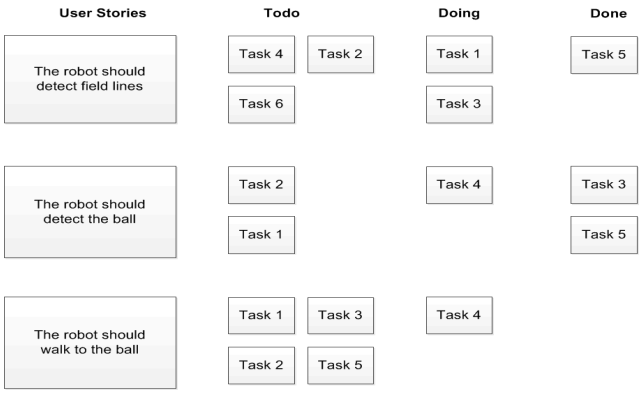

Fig. 10. Example of a Scrum Board

Additionally to the more reliable status, the fact that the students had to explain their progress to their friends in the group led to a better commitment of the students that work voluntarily for their projects. These positive effects were also described in the case studies about agile methodologies in a teaching context.

Beside these positive effects of Scrum we still perceive the following issues. The role of the Product Owner is still open. This role is in some ways taken by the student group itself and there is the final check during the competition. But an accepted Product Owner outside the group would help to lead the development to the features needed in the competition.

Another issue is testing and quality assurance. In a robotics project automated tests are only possible if there exists a simulation environment. A test with robots is always very resource consuming. Concerning quantity and type of tests, no clear vote could be extracted from the answers in the questionnaire. The issues concerning the Product Owner (resp. Customer in the case XP is used) and testing were also perceived in [15]. In [17] and [20] the instructor is chosen as the Product Owner which is not possible here because the student group should not be guided by an instructor in this setting.

The authors consider carrying out more tests in simulation environments as an analogy to regression testing during continuous integration in software engineering and integrating regular testing of the competition requirements at least at the end of each Sprint. It also needs to be investigated how to reduce the "cost" of testing and how to reach a mindset that test is as important as development and that it is valuable to invest time for testing.

Additionally, a more in-deep investigation of the alteration of hardware over the time is required, because this has an immediate influence on the predictability of effort for user stories and the quality of the resulty.

7. Conclusion and Future Work

Competitions offer a motivating environment for student-driven projects. The higher the complexity of this environment, the more crucial are the project management methodologies used - not only to ensure good results in the competition, but also to preserve the motivation of the student groups. Here, traditional project management approaches like the waterfall model or the V-model failed.

Agile methods enabled the student group to manage a number of aspects of a complex project in a self-organised way. By introducing basic elements of Scrum, the student group got more control over the project. Starting from this baseline the student group discussed and decided how to address the issues experienced. Hence the student group was enabled to adjust their own way of work to the needs they perceive. We feel that this is an important step to tackle the complexity of advanced robotics competitions while keeping alive the concept of self-organised student groups.

However, there still are a number of open ques-

tions, minor shortcomings and incompatibilities between Scrum and a self-organised system development which are mainly the length of Sprints, breakdown of User Stories, the role of the Product Owner, testing and quality assurance and the alteration of hardware over time. Based on the experiences and ideas presented here, these issues will be further investigated in order to allow student groups to shape their own agile methodology based on Scrum.

AUTHORS

Reinhard Gerndt* – Ostfalia University of Applied Sciences, Wolfenbuettel, Germany, Wolfenbuettel, Germany, e-mail: r.gerndt@ostfalia.de.
Ina Schiering – Ostfalia University of Applied Sciences, Wolfenbuettel, Germany, Wolfenbuettel, Germany, e-mail: i.schiering@ostfalia.de.
Jens Lüssem – University of Applied Sciences Kiel, Kiel, Germany, e-mail: jens.luessem@fh-kiel.de.
*Corresponding author

REFERENCES

[1] "Jenkins". http://jenkins-ci.org/.

[2] "Redmine". http://www.redmine.org/.

[3] "Robocup". http://www.robocup.org/.

[4] "Robocup - soccer humanoid league". http://www.robocup.org/robocup-soccer/humanoid/.

[5] "Robocup@work". http://www.robocupatwork.org/.

[6] "Robot operating system". http://www.ros.org/.

[7] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al., "The agile manifesto", *http://www. agilemanifesto. org/principles. html. Acesso em*, vol. 7, no. 08, 2001, p. 2009.

[8] J. Dessimoz, ed., *Cognitics*, Roboptics Editionsr, 2011.

[9] J. Dewey, *Experience and Education*, Touchstone, 1938.

[10] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review", *Information and software technology*, vol. 50, no. 9, 2008, pp. 833–859.

[11] R. Gerndt, M. Bohnen, R. da Silva Guerra, and M. Asada. "The robocup mixed reality league - a case study". In: *The Engineering of Mixed Reality Systems*, pp. 399–418. 2010.

[12] R. Gerndt and J. Lüssem, "Mixed-reality robotics - a coherent teaching framework". In: R. Stelzer and K. Jafarmadar, eds., *Proceedings of 2nd International Conference on Robotics in Education (RiE 2011)*, vol. 1, 2011, pp. 193–200.

[13] K. S. J. Sutherland. "The scrum papers: Nuts, bolts, and origins of an agile process", 2007.

[14] C. A. Kent Beck, *Extreme Programming Explained: Embrace Change (2nd Edition)*, Addison-Wesley Professional, 2004.

[15] P. Lappo, "No pain, no xp observations on teaching and mentoring extreme programming to university students", *Agile Aliance*, vol. 1, 2002.

[16] J. Lüssem, F. Pavkovic, U. Samberg, and A. Struck, "Combining learning paradigms to ensure successful learning outcomes in the aera of sotware development". In: *Proceedings of 3rd International Conference on Education and New Learning Technologies (EDULEARN 2011)*, vol. 1, 2011, pp. 81–87.

[17] L. Pinto, R. Rosa, C. Pacheco, C. Xavier, R. Barreto, V. Lucena, M. Caxias, and C. Figueiredo, "On the use of scrum for the management of practcal projects in graduate courses". In: *Frontiers in Education Conference, 2009. FIE '09. 39th IEEE*, vol. 1, 2009, pp. 1–6.

[18] D. Sanders, "Using scrum to manage student projects", *J. Comput. Sci. Coll.*, vol. 23, no. 1, 2007, pp. 79–79.

[19] K. Schwaber, "Scrum development process". In: *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications (OOPSLA)*, vol. 1, 1995, pp. 117–134.

[20] A. Shukla and L. Williams, "Adapting extreme programming for a core software engineering course". In: *Software Engineering Education and Training, 2002. (CSEE T 2002). Proceedings. 15th Conference on*, vol. 1, 2002, pp. 184–191.