07-01-2013

Deliverable DS4.1.3: Report on Summer School for Developers

Deliverable DS4.1.3

Contractual Date:	30-11-2012
Actual Date:	07-01-2013
Grant Agreement No.:	238875
Activity:	SA4
Task Item:	Task 1
Nature of Deliverable:	R (Report)
Dissemination Level:	PU (Public)
Lead Partner:	AMRES
Document Code:	GN3-12-442
Authors:	C. Mazurek (PIONIER), B. Marović (AMRES), M. Lewandowski (PIONIER), P. Kędziora (PIONIER)

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7 2007–2013) under Grant Agreement No. 238875 (GÉANT).

Abstract

This document reports on the Summer School for Developers 2012, covering general event information, event structure, the scope of workshops and coding sessions. The report also summarises attendee feedback and results.

Table of Contents

Table of Figures

Executive Summary

This document reports on the GN3 Summer Developers' School (SDS) 2012 held in Krakow, Poland, which was the third summer school event that SA4 (Software Governance) Task 1 (Best Practices) organised for GN3 software developers. The objectives of the SDS 2012 were to:

- Convey technical knowledge about software development approaches.
- Provide hands-on coding experience.
- Facilitate an exchange of expertise and software development good practices.
- Improve the confidence of development teams when applying development processes.
- Leverage the quality of GN3 software and services.

The main aim of the SDS 2012 was to popularise continuous delivery (CD) mechanisms, testing practices and Agile Programming techniques among GN3 software developers. This report details the CD workshop, Testing session and DevOps session. It also describes and summarises the Behaviour-Driven Development (BDD) [**BDD**] coding sessions, where attendees were given tasks to be solved using Agile Programming practices. These sessions were based on the programming tools already used in GN3, and included use of GN3 development infrastructure. The event ended with a workshop, where all participating teams presented their solutions to coding tasks, which were discussed and evaluated.

The results of a survey conducted to evaluate the content and organisational aspects of SDS 2012 are also provided in this report. It also assesses the event's influence on the day-to-day work of GN3 developers, the importance of testing activities, and its positive impact on code quality.

1 Introduction

Summer Developers School 2012 (SDS 2012) was the third occurrence of an annual event, which aimed to share the theoretical knowledge and hands-on software engineering expertise of developers involved in GN3 Service (SAs) and Joint Research (JRAs) Activities. The event was held between 3 and 7 September 2012, at the AGH University of Science and Technology in Kraków, Poland. The main topics covered were related to the software continuous integration and delivery, as well as testing approaches and DevOps methodology. The event also aimed to foster the adoption of SA4-T3 software development and QA testbed infrastructure among GN3 developers.

Organised by SA4 Software Governance Task 1, the Summer School was attended by developers from AMRES, GRNET and PSNC, representing AutoBAHN, cNIS, GENUS and perfSONAR teams.

The Summer School for Developers 2012 included the following sessions:

- A one-day workshop: “Continuous delivery of software”, focused on aspects of continuous integration and deployment, especially S/W tools and techniques used while developing Java software.
- Testing tools and approaches for software development: a half-day session on S/W tools, mainly for unit and integration testing.
- A two-hour session on DevOps Agile methodology and its potential application within GN3 (“Towards DevOps methodology”).
- Hands-on coding sessions, including the ‘Coderetreat’, a day-long, intensive practice event, focusing on the fundamentals of software development and design and the Problem-solving day, which worked on troubleshooting and implementing programming tasks, with awards given for the top-three solutions.

An agenda for SDS 2012 is provided in Appendix A. The following chapters provide a detailed description of SDS 2012, as well as the expected impact on the GN3 project.

2 SDS 2012 Structure and Scope

2.1 Continuous Integration Workshop

The goal of this workshop was to introduce and provide examples of continuous integration (CI), and eventually, continuous delivery (CD) in software development. A thorough description of the process was presented, including the open source tools that can support development. Sessions were led by an external expert, Tomasz Kaczanowski [**TKaczanowski**], who is an author of "Practical Unit Testing with TestNG and Mockito" [**UnitTestNGMockito**], TDD (Test-Driven Development) enthusiast and open source proponent. Kaczanowski is a senior Java developer and technical team lead, with a primary focus on code quality, testing and automation.

The workshop explained the purpose of introducing continuous integration in software development, its benefits and related costs, especially when the work with CI begins. While the costs are mostly visible during the work-intensive setup of the infrastructure and the change of QA personnel work attitude, the described benefits are mainly related to reduction of risks (since there are no large code merges required), improved team organisation, rapid feedback and better visibility. The course also elaborated on the differences between continuous integration, delivery and deployment. A practical approach to code structure, highlighting the creation of branches and feature toggles was also presented.

For each step in the CI process, a supporting software tool was presented in order to automate as many operations as possible. Attendees were already familiar with Maven [**Maven**], so only advanced aspects related to profiles and plugins, snapshots and releases were presented. Flyway [**Flyway**] was introduced as a tool to support database schema versioning, which featured a discussion of rollback issues. In turn, the Jenkins continuous integration server was explained in terms of parameterised jobs, the jobs matrix, as well as project-build pipelines. The workshop also included a summary of best and worst practices related to the CI process.

Software tools supporting code quality checks were also introduced, in particular, Integrated Development Environment (IDE) tools and standalone tools that could be integrated with the CI server. PMD [**PMD**], Checkstyle [**Check**] and Findbugs [**FindBugs**] were described as tools for integration with IDE, whereas Sonar [**Sonar**], was used as an example of a quality platform that can be integrated with Jenkins to become a code quality dashboard in the CI process. Although using software tools for code quality is recommended, the speaker also pointed out the downside of relaying on tools only. Hence, the tools presentation also described the benefit of code reviewing and pair programming as activities that have a direct impact on code quality.

2.2 Testing Tools and Approaches for Java Developers

During the second day of the SDS, a selection of tools supporting Java developers' software testing activities, as well as good practice for writing tests were presented. The session covered testing tools such as Awaitility [**Await**] (testing asynchronous operations), Catch-Exception [**CatchExc**] and Fest Fluent Assertions [**FestAssert**]. The speaker also introduced the advanced aspects of using a TestNG [**TestNG**] framework, comparing it to a JUnit [**JUnit**] framework. The focus of the session was on concurrency and parameterised

tests, test groups and test dependencies. Good testing practices were also summarised, and examples of what values shall be tested were provided. The session ended with a quiz, based on good and bad examples of tests from the day-to-day work of software developers.

In terms of testing approaches, the appropriate use of Test and Behaviour-Driven Development was presented, including how to start with this approach and highlighting when it should not be used.

2.3 Towards DevOps Methodology

The collaboration between software development and IT operations (and communication between the respective teams) is one of the key aspects of deploying and releasing software products. DevOps is one of the methodologies used to combine software development and IT operations. This methodology is even more valid within the context of the GN3 project, as many software products are in transition towards production environments. DevOps can also help to minimise risky deployments, as it integrates with continuous delivery and frequent releasing.

The DevOps methodology was presented by Rade Martinović (AMRES/UoB), who explained its origins, motivation and identified sample projects where it worked well. Specialisation between development and operations is often desirable, and indeed inevitable, but development and operations teams can often have conflicting goals, as developers wish to rapidly deliver new features, while operators are required to keep the systems stable and reliable. Developers are often seen as feature-driven, freewheeling, collaborative, and self-organising, while operators are perceived to be driven by operational policy and a command-and-control attitude focused on uptime and compliance. This difference leads to isolation between project groups, and may greatly affect efficiency and overall service quality. As a result, the session on DevOps described the methodology both from the perspective of software developers and IT operations. The need for IT operations to place greater emphasis on automation within their management of infrastructures, changes, configuration, as well as service and platform provisioning was stressed. It is also important that IT specialists and system administrators are familiar with new tools and the code itself. The development teams should be given the option to deploy their own code and be able to handle resources/infrastructures on a self-service basis (testing, developing and managing pre-production environments based on cloud solutions). This requires developers to learn how the IT operations infrastructure is managed, and how and where the actual deployment is performed. The importance of communication and collaboration was also highlighted, since the common goal for both teams is to deliver value to the customer. The speaker also presented the benefits of introducing DevOps and the possibilities of arranging related infrastructures in cloud-based environments.

2.4 Coding Sessions Based on Agile Practices

2.4.1 Coderetreat

During SDS 2012, one of the hands-on coding sessions was held in a form of “Coderetreat”, inspired by a formula of day-long workshops for programmers set up to provide intensive skills practice, away from the pressures of the day-to-day environment.

Coderetreat provided a day of intensive software development and design practice, examining basic coding principles, object-oriented design, modularity, and proper functional testing. By providing developers the opportunity to take part in focused practice, away from the pressures of 'getting things done', this format has proven itself to be a highly effective means of skill improvement. By practicing the basic principles of modular and object-oriented design, developers were able to improve their ability to write code that minimises the cost of change over time.

The event structure is an established and time-tested format, which helps attendees focus on software development practices while solving a main task. The day was divided into 5, 45-minute sessions, with each session ending with a recap of the activity and short break. Since pair programming is a core principle of Coderetreat, attendees participated in pairs, which changed partners after each of the three morning sessions. In addition, the code was to be deleted after each session, as each session's learning built upon the previous sessions. During the morning, attendees focused on becoming comfortable with the problem to be solved and breaking old software development habits. The two afternoon sessions challenged teams with new constraints and rules, which expanded their skills and understanding of abstractions, modular design and Behaviour-Driven Development. The four rules of simple design (passing tests, no code duplication, and clear file naming and keeping the code concise) were followed during each session.

The Coderetreat held at SDS 2012 was also enriched by the introduction of a number of constraints relating to the way attendees implemented their solutions. During each session, developers were given new rules to follow, related to not using primitive types and arrays, silent coding (communication only through code), limited methods' length and BDD approach. The adoption of every constraint was discussed during the session recaps.

During Coderetreat, the attendees were given the "Convay Game of Life" problem to be implemented [**Convay**]. However, the selection of the problem was open, and dependent on the implementer's decision. Solving the problem itself is only a background task, while the true aim of the day is to practice fundamentals of software design and development, leveraging benefits of pair programming and stimulating adoption of good programming techniques and advanced skill development.

2.4.2 Main Coding Task

As the Summer School is an event combining lectures and hands-on sessions, programming skills are developed and good practices conveyed as part of the main coding task. The positive adoption of Test-Driven Development during SDS 2012 built upon those experiences, and the requirements of main coding task were defined in a form of user stories to be implemented using the BDD approach. User stories described features of a hypothetical London Olympic Games web application for calculating games' statistics.

The structure of a coding task defined three, 90-minute sessions, where each session contained new user stories to be covered. The user stories were introduced gradually, and together, they helped to build the set of features for a final web application (the deployable application to the web server, with a user interface that would allow execution of functionalities covered in the user stories). While developing solutions, attendees were encouraged to put some Agile practices in action. In particular, pair programming was one of the developers' practiced techniques. It turned out to be beneficial from code reviewing and code-quality perspective.

Behaviour-driven development was also an obligatory approach, what highlighted the importance of writing tests.

The task solutions were developed in teams of pairs, and evaluated according to the following weighted criteria:

- Completeness of implemented user stories (5 points, with a weighted score of 3 = 15 possible points).
- Test coverage (branch coverage) (5 points, with a weighted score of 3 = 15 possible points).
- Appliance of BDD (5 points, with a weighted score of 3 = 15 possible points).
- Whether the application can be built and is deployable (5 points, with a weighted score of 2 = 10 possible points).
- Code violations (based on PMD, Checkstyle and Findbugs reports) (5 points).
- Use of mocking objects to simulate database interaction and simplify testing (5 points).
- Use of Subversion and Jenkins CI (hosted by SA4-T3 S/W development infrastructure) (5 points, with a weighted score of 2 = 10 possible points).

The evaluation process was carried out by the organisers from SA4, and the overall scoring showed that the standard of the proposed solutions was very even. Hence, the code quality platform for static code analysis, Sonar, was used for calculating test coverage and code violations. Sonar helped to precisely assess the value of test coverage and code violations. The differences between team scores were small, however, the three best solutions were selected and presented with an award during the event's closing session.

3 Aspects of the SDS that are beneficial for the GN3 Project

3.1 Behaviour-Driven Development (BDD)

Behaviour-Driven Development was introduced during SDS 2011, and due to excellent take-up, it was implemented during SDS 2012 in many hands-on exercises. BDD principles were outlined during the retrospective session and used in both the Coderetreat event and in the main coding task. During the Coderetreat, BDD was meant to be a constraint, however, attendees had used this approach since the very first session as the driving technique for problem solving. In the main coding task, BDD was a requirement and one of the criteria when evaluating solutions. When applying BDD, the attendees were given the functional requirements defined in the form of user stories. The complexity of following user stories was increased with each session. While implementing test cases, developers used a 'given-when-then' pattern to structure their approach and improve tests' readability. The use of BDD had another consequence, however, as the resulting object model of implemented solutions was strictly related to the defined user stories (and test cases). The fact of creating each class was derived from the concrete test case, eliminating unnecessary code. In turn, the BDD approach led to high code coverage (branch coverage), which increased the overall quality of written code.

The application of BDD was widely taken up by participants, as they decided to use this technique even during the Coderetreat sessions, although it was only required during the last session. The attendees learned that the test-first approach provided constant feedback on the correctness of their code and improved the design, as they used their own code base. Hands-on exercises also provided a chance to introduce and execute other frameworks, which support developers in their daily work. The use of open source libraries, such as FestAssert [**FestAssert**] and Mockito [**Mockito**] were recommended to improve the overall readability of test code and spare developers additional effort. In some cases, such tools were necessary for proper implementation of solutions (e.g. mocking objects).

3.2 Software Continuous Delivery

Continuous integration is already present in GN3 software developments, and the use of the Jenkins CI server is periodically verified by QA Personnel during audits. For more than ten GN3 software projects, the project build process has been controlled by the CI server. During SDS 2012, the extension of CI process towards continuous delivery (CD) was presented. The speaker explained the process from the commit stage, through to many tests and then to software packaging, where the final software build is released and ready for deployment. SA4 Software Governance delivers a development infrastructure that includes the GN3 Jenkins CI server [**GN3Jenkins**], as well as a repository for software artifacts (Artifactory [**Artifactory**]). The workshop was an opportunity to stress the importance of using proper S/W tools in continuous delivery process to reach a high level of automation. It raised GN3 developers' awareness that the provided infrastructure already meets requirements to support continuous delivery. Some GN3 software developments (e.g. cNIS team) already execute the CD process using provided support tools, however, this approach needs more attention more promotion and propagation within the project.

The workshop on continuous delivery also provided a good occasion to promote other tools supporting software development. As software quality control plays important role in SW development, the Sonar quality platform was introduced to inform attendees how to monitor the quality of project builds in a unified way.

3.3 Towards DevOps in GÉANT

Currently, some software developments in the GN3 project are in a transitional phase from development to production. In order to present a methodology on how to manage the communication and collaboration between development teams and IT operations, the DevOps session was held. Applying DevOps brings potential benefits for the project and, in particular, delivery of software products, as this methodology enables rapid feedback on potential flaws that have not been detected during testing activities. It also shortens the feedback loop, which reduces the stress of the release and deployment process. A seamless transition between development and operations also has an impact on the first-to-market factor.

DevOps also introduces the concept of environments to support the software development and release process. Building upon the S/W Development and QA Testbed infrastructures delivered by SA4, GN3 development teams can introduce staged environments for developing and testing their software products. Following the DevOps paradigm, it is recommended to build environments for development, system integration testing, user

acceptance testing, staging, production and system volume testing. QA Testbeds are already used by GN3 development teams to set up testing and demonstration instances of developed software, hence it could be extended to align with DevOps best practices.

3.4 Extreme Programming Practices

The Summer School for Developers presented a rare opportunity to train developers in new techniques that they can then apply to their day-to-day work. Since the event advocates Agile Practices in action, pair programming was selected to be the driving technique for both the Coderetreat and main coding task. It is an approach in which two programmers work together at one workstation, with one playing a role of a driver (who writes the code) and the other an observer who reviews the code as it is typed in. The roles are switched often. This technique is one of the most efficient in terms of code quality, and is considered as good code review practice. Arranging work in such a way results in shorter, better designed programs with fewer flaws. The other reason behind the introduction of pair programming was the efficient knowledge sharing, which is one of core principles of SDS events. The exercise intensified as the pairs were swapped at each session during the Coderetreat day. The experience of programming with many people delivered opportunities to improve self-discipline, stay focused on the solution and gradually improve the final implementation of the coding task.

4 Survey Results

During the closing session of the SDS, the 13 attendees were asked to fill in a non-anonymous survey to provide feedback on the content of workshop and coding sessions, as well as on event logistics. The aggregated results are detailed below.

4.1 Workshop and Coding Sessions

The following charts illustrate the aggregated feedback addressing the evaluation of event topics.

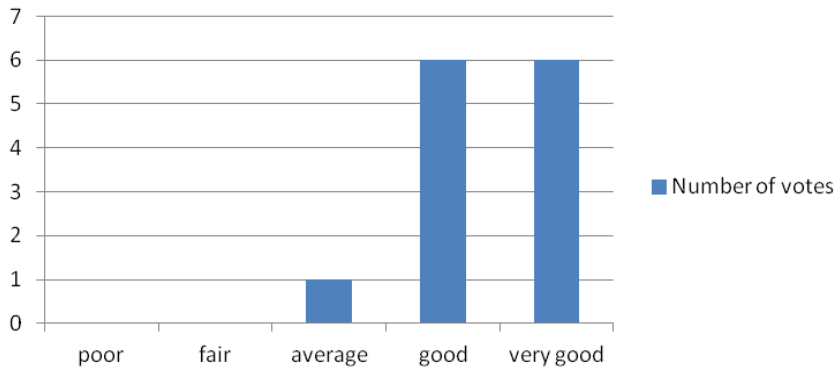

Figure 1: Evaluation of tutorial topics

Overall, 92% of attendees found the range of tutorial topics to be “good” (46%) or “very good” (46%), and 8% marked it as “average”.

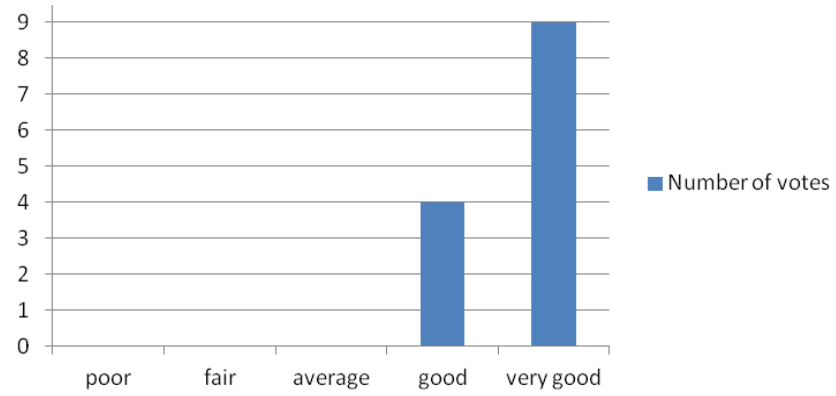

Figure 2: Coding session’s topic evaluation

The attendees found the SDS 2012 coding sessions to be “very good” (70%) or “good” (30%).

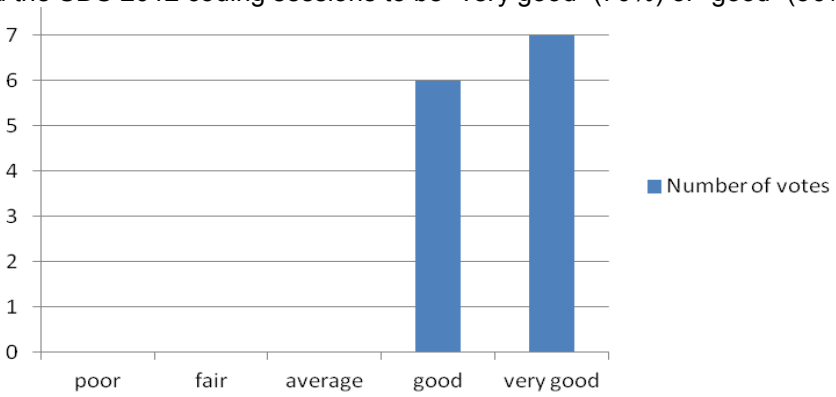

Figure 3: Knowledge of lecturers

The attendees evaluated the lecturers’ knowledge as “very good” (54%) or “good” (46%).

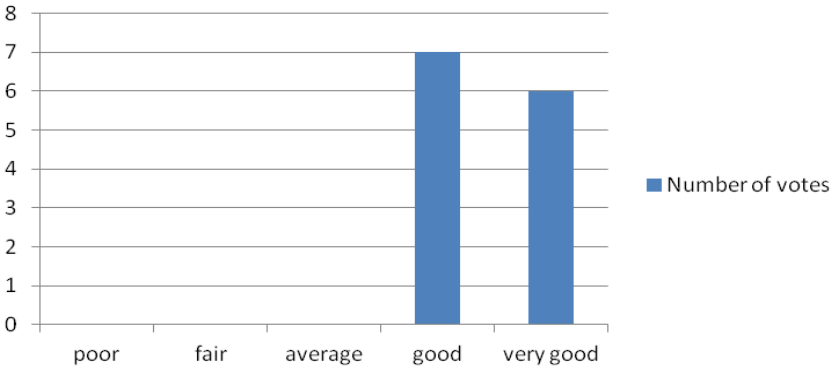

Figure 4: The value of information and experience

The attendees evaluated the value of received information and gained experience as “very good” (46%) or “good” (54%).

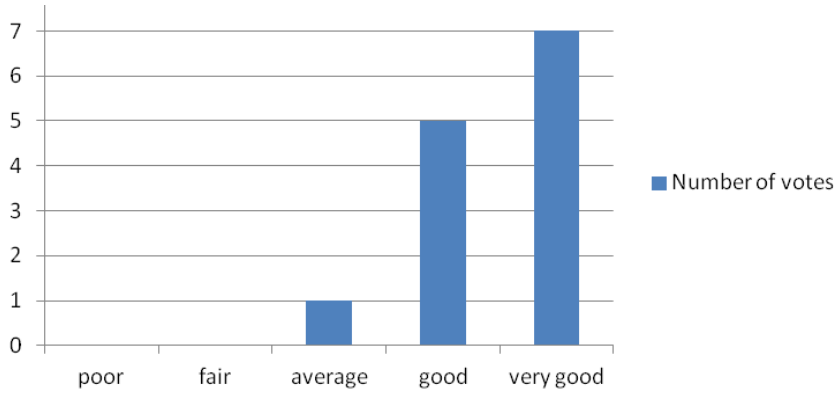

Figure 5: Usefulness of provided materials

A total of 92% of the attendees marked the usefulness of the materials provided during the event as “very good” (54%) or “good” (38%). One person (8%) rated the materials as “average”.

4.2 Organisational Aspects

The following charts illustrate the aggregated feedback addressing the organisational aspects of the event.

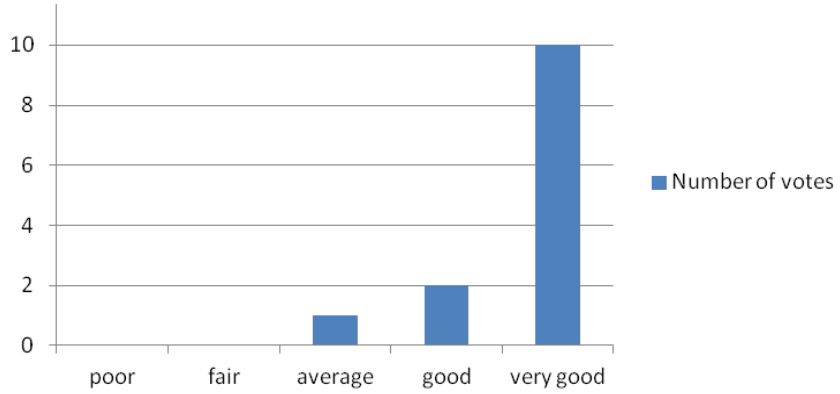

Figure 6: Accommodation and event facilities

A total of 92% of attendees evaluated the accommodation and event facilities during SDS 2012 as “very good” (77%) or “good” (15%). Only one person (8%) found it to be “average”.

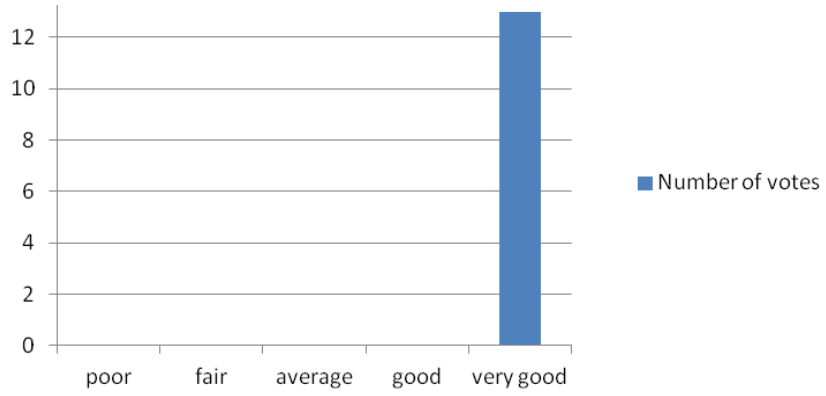

Figure 7: Event organisation

Organisation of SDS 2012 was unanimously (100%) rated as “very good”.

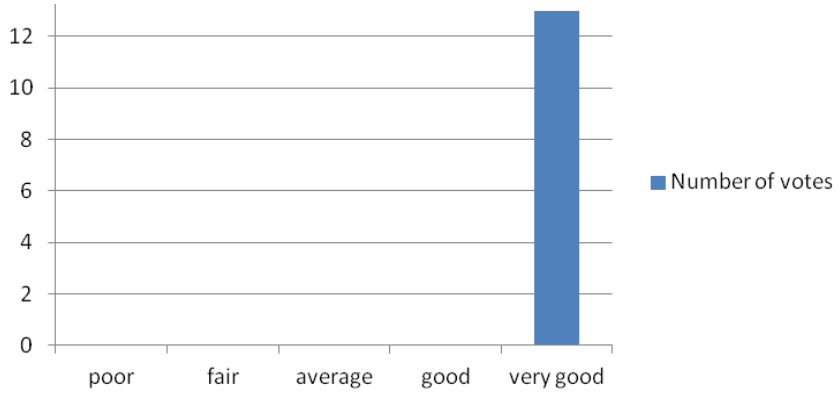

Figure 8: Evaluation of the social event

The networking activity provided by the event organisers was unanimously (100%) ranked as “very good”.

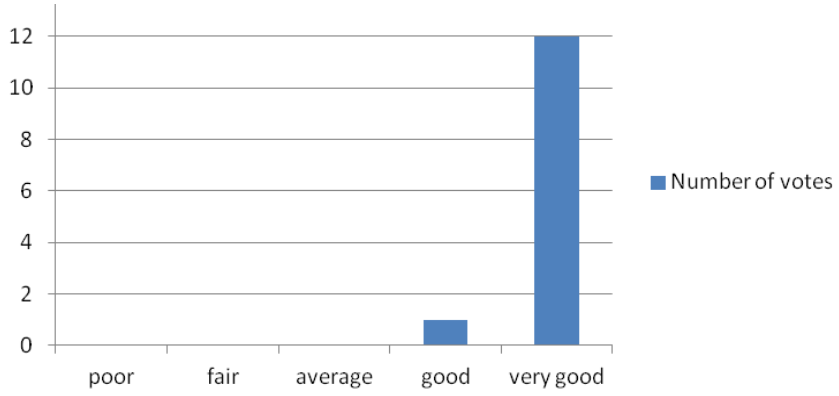

Figure 9: Social atmosphere at the event

100% of the attendees found the social atmosphere at the event to be “very good” (92%) or “good” (8%).

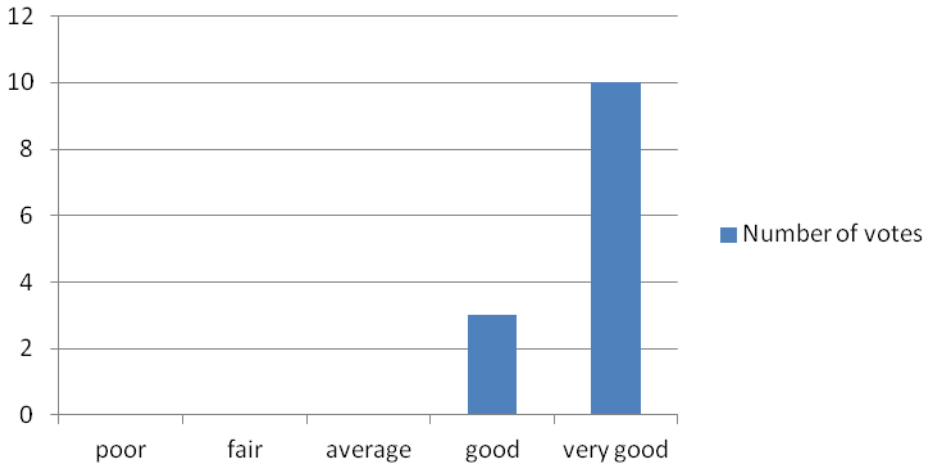

Figure 10: Overall rating of SDS 2012

100% of attendees provided the overall rating of the event “very good” (77%) or “good” (23%). This rating was based on a direct question included in the attendees’ survey.

4.3 SDS 2010, 2011 and 2012 Survey Results Comparison

The following charts illustrate how organisational issues and workshop/coding sessions were evaluated by SDS attendees in 2010, 2011 and 2012. In general, there is clear improvement regarding the effectiveness of the course (overall ratings can be seen to improve over time). Although the SDS 2011 tutorial sessions received the most positive feedback, attendees found the SDS 2012 coding sessions to be the most valuable. Within all three editions of the event, the value of information and experience gained during SDS increases (to the point of receiving only "good" and "very good" notes in 2012).

The organisational aspects of the Summer Developers' School sessions have always been very highly rated (receiving "good" and "very good" responses).

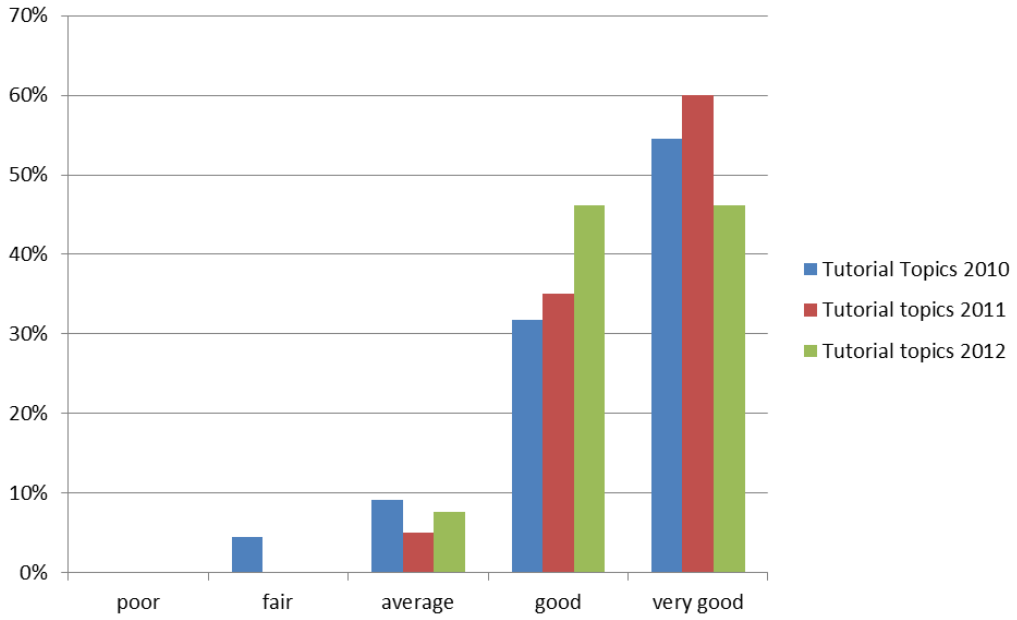

Figure 11: Tutorial topics 2010, 2011, 2012

In 2010, 85% of attendees ranked tutorial topics as "good" or "very good", 9% found it to be "average", and 4% rated it as "fair". Attendees found SDS 2011 tutorial topics to be the best (95% "good" or "very good"). In 2012, 92% of questioned developers awarded "good" or "very good" scores. In 2011 and 2012, there were neither "fair" nor "poor" marks.

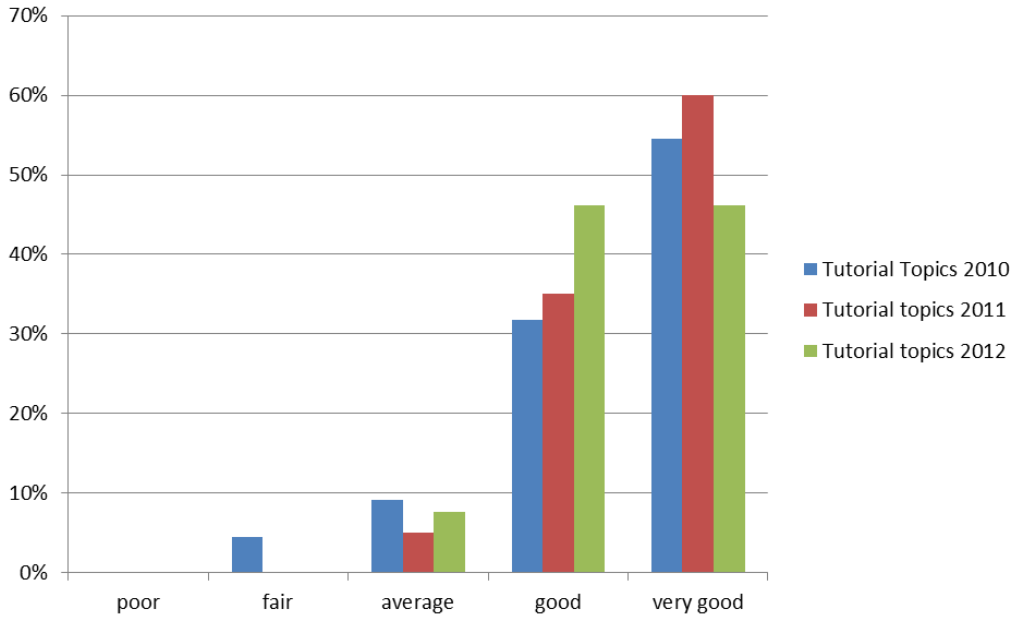

Figure 12: Coding sessions' topics 2010, 2011, 2012

Inevitably, SDS 2012 provided the most valuable coding sessions (100% of "good" and "very good" notes). SDS 2011 received the highest percentage of "fair" notes (15%).

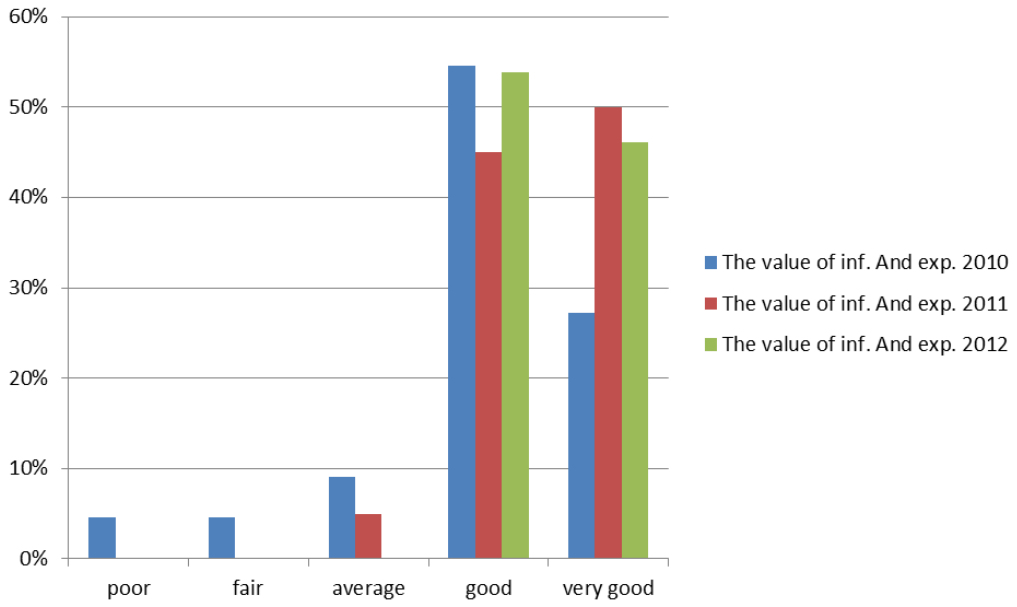

Figure 13: The value of information experience 2010, 2011, 2012

Similarly, the value of information and experience provided during SDS 2012 was marked as "good" or "very good" by 100% of developers. In 2011, 5% of responders claimed that it was "average". After SDS 2010, 9% of developers were not satisfied with the provided information and experience gained (4.5% rated it as "poor" and 4.5% as "fair").

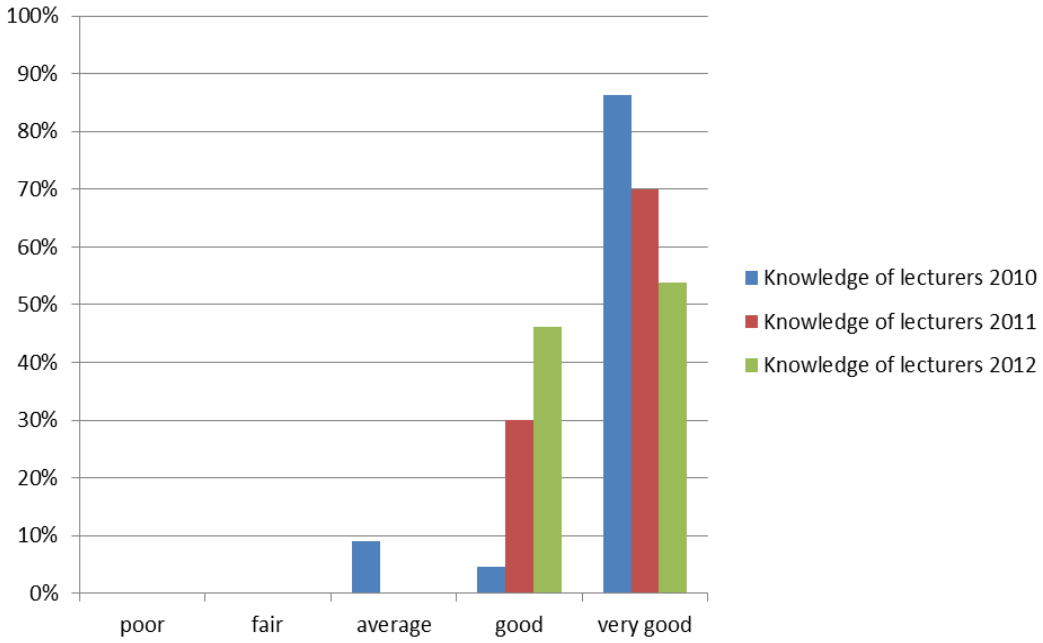

Figure 14: Knowledge of lecturers 2010, 2011, 2012

In 2012, 100% of developers evaluated the knowledge of lecturers as "good" or "very good". Generally speaking, SDS attendees appreciated lecturers' knowledge, "average" is the lowest mark that has ever been given (during the first SDS in 2010).

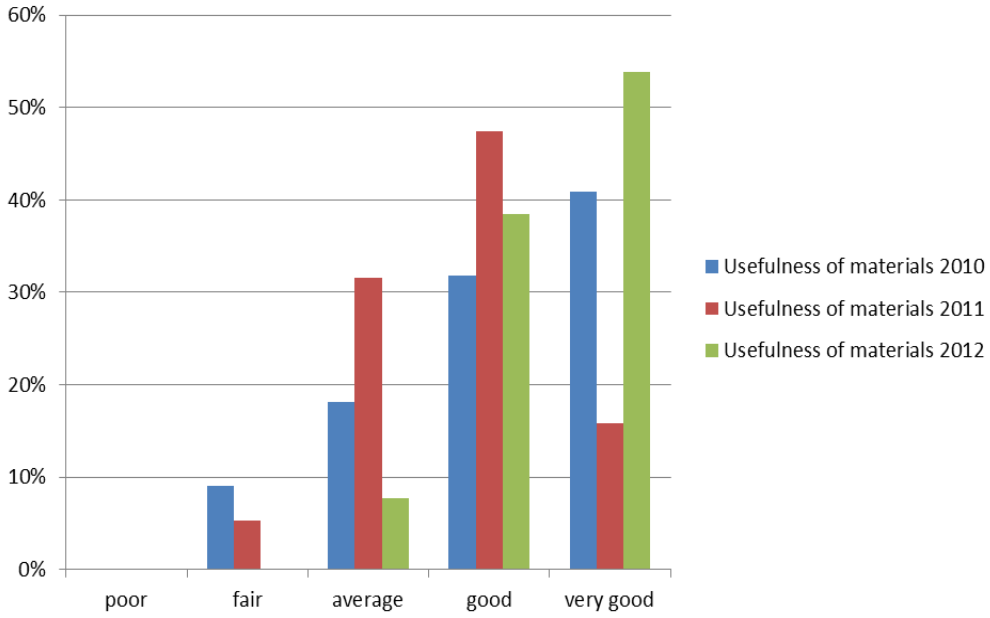

Figure 15: Usefulness of provided materials 2010, 2011, 2012

In opposition to SDS 2010 and SDS 2011, attendees of SDS 2012 rated the usefulness of provided materials neither "poor" nor "fair". Also number of "average" marks was significantly reduced (from 31% in 2011 to 7% in 2012).

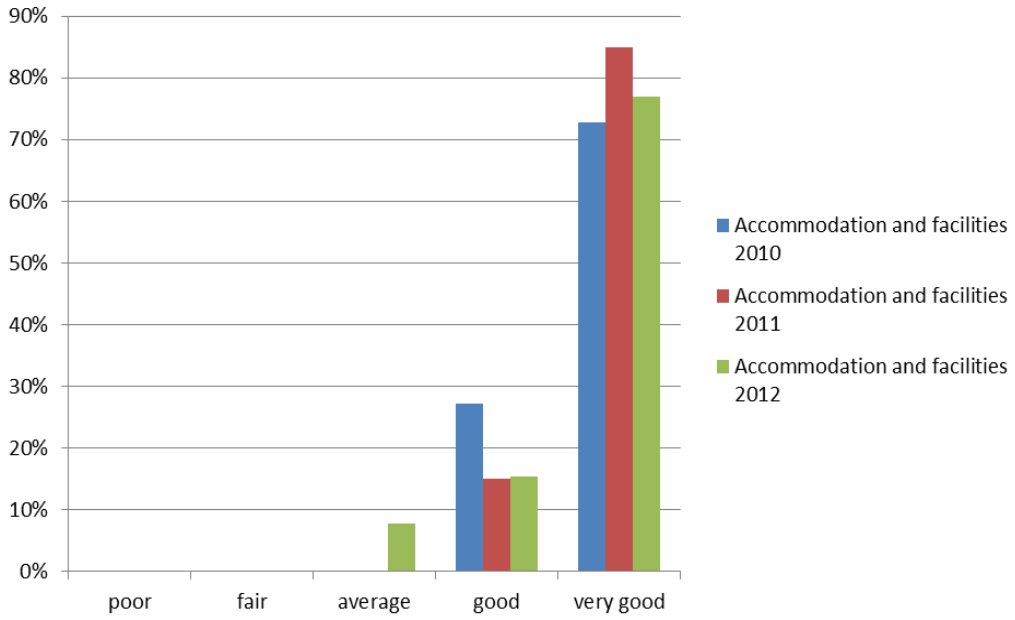

Figure 16: Accommodation and facilities 2010, 2011, 2012

Only one person found the SDS 2012 accommodation and facilities as “average”. In 2010 and 2011 100% of developers claimed it was “good” or “very good”.

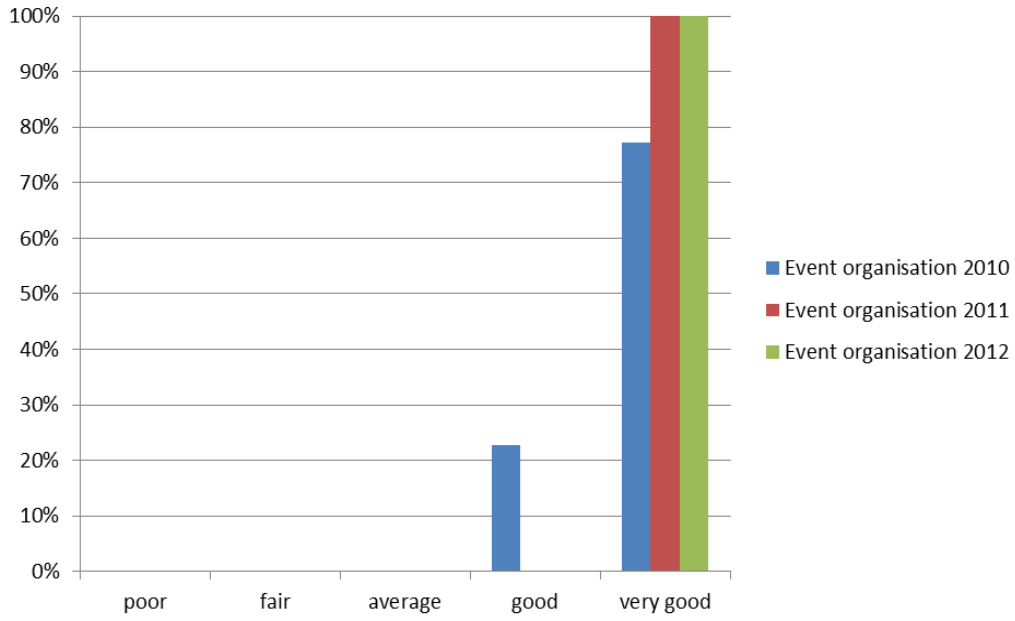

Figure 17: Event organisation 2010, 2011, 2012

Lessons learnt by organisers in 2010 resulted in improved organisation of SDS 2011 and SDS 2012 (100% of “very good” ratings for both events).

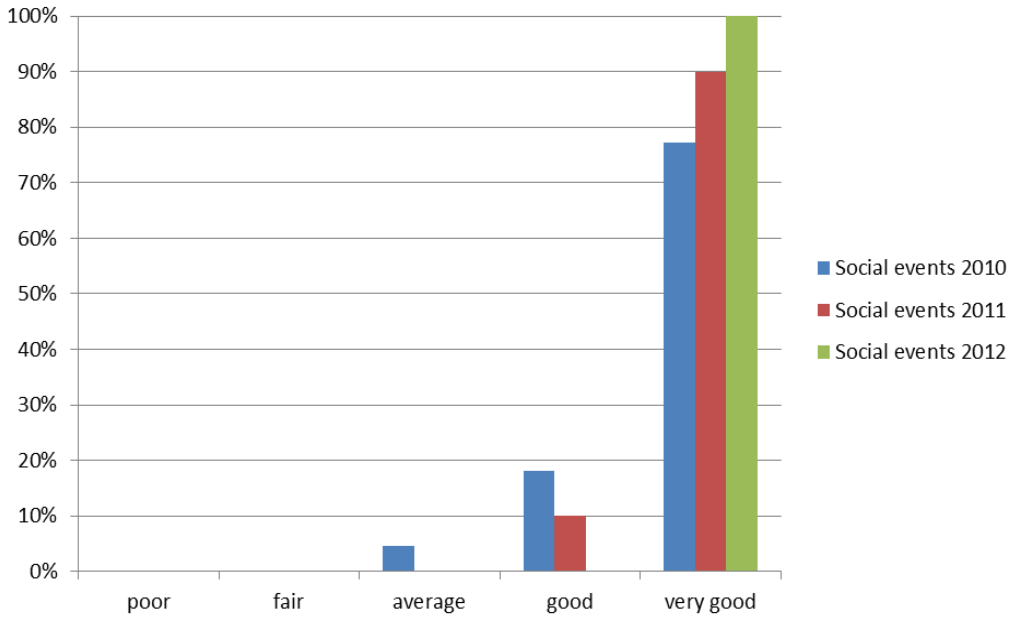

Figure 18: Social events 2010, 2011, 2012

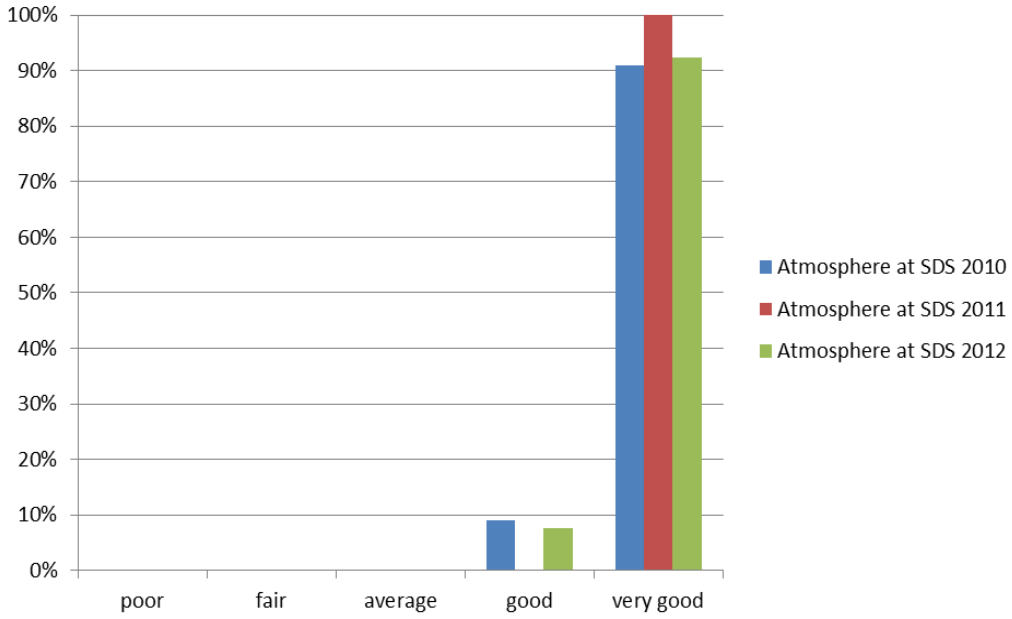

Figure 19: Atmosphere at SDS 2010, 2011, 2012

Developers felt very comfortable during Summer Developers' School. Atmosphere during three series was marked as "good" or "very good".

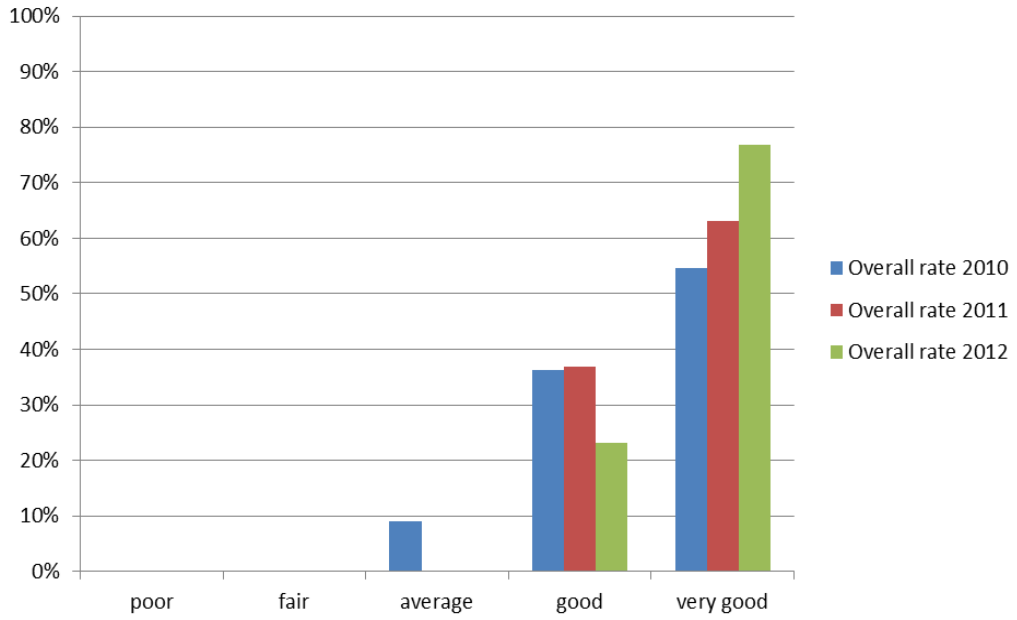

Figure 20: Overall rating for 2010, 2011, 2012

The overall rating reflects how Summer Developer Schools were received by attendees: while in 2011, 37% of attendees marked it as “good” and 63% as “very good”, the numbers in 2012 are 23% and 77%, respectively.

5 Conclusion

The Summer School for Developers 2012 aimed to improve technical knowledge of GN3 developers by exposing them to workshops comprising hands-on problem solving and lectures. This year, the event focussed on popularising continuous delivery, Behaviour-Driven Development, DevOps methodology and Agile programming practices. SDS 2012 met the objectives, as determined by the survey results carried out among attendees. In particular, it was stated that the main coding task should be more general and based on frameworks that are commonly used in industrial applications. Results of the survey carried out among SDS 2012 attendees indicated, that all coding sessions (both the Coderetreat and the main task) were well-accepted by developers and met their requirements (100% of attendees rated them either “good” or “very good”).

Developers found that SDS 2012 provided a great opportunity to share and exchange their skills, knowledge and experience (all rated the event either “good” or “very good”), and expressed their desire to participate in future Summer Schools, if such workshops are organised. Thus, it is very important that JRAs and SAs activities and task leaders encourage team members to join the SDS community of developers (if this event is to be continued as part of GN3plus), so they can learn new skills and techniques that will have a positive impact on their daily work. One cannot underestimate the networking aspects of Summer Schools – developers that know each other from e-mail conversations or videoconferences can meet together and help build team spirit. This aspect was also kindly welcomed by developers who, according to latest survey, unanimously ranked the SDS 2012 atmosphere as “very good” (93%) and “good” (7%). This year, SDS 2012 was also described in an article in GÉANT CONNECT magazine issue #9 [**CONNECT**], where the opinions of attendees were presented as a series of brief interviews.

Attendees of the SDS series all agreed that the knowledge and skills gained during Summer Schools increased their effectiveness, which is apparent in the higher quality of the software that they develop. As supported by comments from the GN3 review, SDS is clearly a beneficial and worthwhile event, for both developers and the project.

Appendix A Summer School for Developers 2012 Programme

Day 1 (3 September)
Towards Continuous Deployment

9:00–9:30 Registration
9:30–10:00 Opening
10:00–11:30 Continuous delivery (workshop)
11:30–12:00 Coffee break
12:00–14:00 Continuous integration/ delivery (workshop)
14:00–15:00 Lunch
15:00–16:00 Code quality (workshop)
19:30 Networking meeting

Day 2 (4 September) Testing

10:00–11:00 Developers tests
11:00–11:30 Coffee break
11:30–13:30 How to write tests
13:30–14:30 Lunch
14:30–15:30 Towards DevOps in GN3
15:30–16:00 Coffee break
16:00–16:30 Summary

Day 3 (5 September) Coderetreat

9:30 – 10:15 Session #1
10:15 – 10:30 Retrospective & pair swapping
10:30 – 11.15 Session #2
11:15 – 11:30 Retrospective & pair swapping
11:30 – 11:45 Coffee break
11:45 – 12.30 Session #3
12:30 – 12:45 Retrospective & pair swapping
13:00 – 14:00 Lunch
14:00 – 14:45 Session #4
14:45 – 15:00 Retrospective & pair swapping
15:00 – 15:15 Coffee break
15:15 – 16:00 Session #5
16:00 – 16:30 Closing cycle

Day 4 (6 September)

9:00–9:30 Briefing
9:30–11:30 Technical session I (coding tasks)
11:30–12:00 Coffee break
12:00–14:00 Technical session II (coding tasks)
14:00–15:00 Lunch
15:00–17:00 Technical session III (coding tasks)

Day 5 (7 September)

9:30–11:00 Summary session I
11:00–11:15 Coffee break
11:15–12:30 Summary session II
12:30–13:00 Prizes and farewell family picture
13:00–14:00 Lunch

References

[Await]	http://code.google.com/p/awaitility/
[Artifactory]	http://www.jfrog.com/
[AutoBAHN]	http://www.geant.net/service/autobahn/pages/home.aspx
[BDD]	Behaviour-Driven Development: http://en.wikipedia.org/wiki/Behavior-driven_development
[CatchExc]	http://code.google.com/p/catch-exception/
[Check]	http://checkstyle.sourceforge.net/
[cNIS]	http://www.geant.net/service/cnis/pages/home.aspx
[Coderetreat]	http://coderetreat.org/facilitating/structure-of-a-coderetreat
[CONNECT]	http://www.geant.net/Media_Centre/connect/Pages/home.aspx
[Convay]	http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
[DevOps]	http://en.wikipedia.org/wiki/DevOps
[FestAssert]	http://code.google.com/p/fest/
[Flyway]	http://code.google.com/p/flyway/
[FindBugs]	http://findbugs.sourceforge.net/
[GENUS]	http://www.geant.net/Research/Future_Network_Research/Pages/CurrentandPotentialUsesofVirtualisation.aspx
[GN3Jenkins]	https://ci.geant.net/jenkins/
[JUnit]	http://www.junit.org/
[Maven]	http://maven.apache.org/
[Mockito]	http://code.google.com/p/mockito/
[perfSONAR]	http://www.geant.net/Services/NetworkPerformanceServices/Pages/perfSONARMDM.aspx
[PMD]	http://pmd.sourceforge.net/
[Sonar]	http://www.sonarsource.org/
[TKaczanowski]	http://kaczanowscy.pl/tomek/about-me
[TestNG]	http://testng.org
[UnitTestNGMockito]	http://practicalunittesting.com/

Glossary

API	Application Programming Interface
BDD	Behaviour-Driven Development
CI	Continuous Integration
CD	Continuous Delivery
Dev	Development
DevOps	Development and Operations
JRA	Joint Research Activity
IDE	Integrated Development Environment
PMD	Programming Mistake Detector (unofficial acronym)
QA	Quality Assurance
SA	Service Activity
SDS	Summer Developers' School
SIT	System Integration Testing
SSH	Secure Shell protocol
SVN	Subversion (software versioning)
SVT	System Volume Testing
S/W	Software
TDD	Test-Driven Development
UAT	User Acceptance Testing