Post-mortem Report: GroupAlarm

DAT255

Group: ASIJGE

Date: 2015-05-29

Team: Alexander Hederstaf

Sebastian Odbjer

Isabel Azcárate

Joakim Sehlstedt

Gabriella Hallams

Emma Wibel

Processes, Practices and Methods

The processes, practices and methods used when developing the application GroupAlarm are presented below.

Scrum

We have worked mainly according to scrum, which is an iterative and step-by-step agile software development methodology to manage the development of our application. The main reason we chose to work with scrum is because customers can change their minds about what they want during a project, and Scrum provides the flexibility needed. Although we did not have a real customer it helped the group when facing changes in the design or development process. This development strategy helped us work as a unit to reach a common goal.

Scrum encourages physical co-location or close online collaboration as well as daily face-to-face communication among all team members. We did not fulfill that part due to the reasons mentioned in our reflections over non process specific decisions.

We worked with agile scaling because of the complex product of the project. We turned our user stories into smaller tasks and divided them between the members of the group to be completed during our weekly sprints. First we constructed a basic alarm and made sure of its functionality before adding more features and network activities.

Collective Code Ownership

We also followed the principle Collective Code Ownership where the entire group was responsible for all code, and all members were allowed to change and upgrade any code. The most important thing was the quality of the complete program. We saw this as a way of helping each other and therefore took no pride in our own code. Before pushing code to GitHub the code was approved by our fellow group members. This was a way of simple testing the programming progress during the development process.

Github

Half of the group members had never used Github before which slowed the work process down due to some learning time and mentoring by the other group members. To shorten the learning time and to avoid future mistakes in the work process we mainly used Github's GUI clients, Github Mac and Github Windows, to push our code to the repository. This way those who were not used to using a terminal could use the GUI and those with more

experience could use whatever they prefered. It became a helpful tool to keep everyone updated about the applications progress and to see the other members contributes.

Google Drive

We used Google Drive for documentation which made it possible for us to work both together and parallel with different tasks on different locations. It has been a great asset to the group.

Android Studio

The application was developed in Android Studio, a program only a few members of the group had experience with. Android Studio is an application development program. The program has steadily been improved and modified during the last couple of years. As a result, knowledge about the program gets outdated fast. This made the programming environment fairly unfamiliar to all of us, which resulted in the startup process taking longer than we had anticipated. We learned how Android Studio works both by listening to each other and studying program tutorials online. Since the online sites providing help for Android are sometimes outdated, you had to make sure that the solution presented was still relevant for the task you were trying to solve. Which resulted in a lot of time being spent searching for solutions. Despite our previous knowledge of the java language, Android SDK took time to learn and made our application development slower and sometimes a bit ineffective.

Weekly sprints and meetings

Once a week the group had a meeting where the goal was to answer three key questions:

1. *What did I accomplish this week?*
2. *What will I do next week?*
3. *What obstacles are impeding my progress?*

This type of meeting is in a professional development process done daily and referred to as “daily scrum” but in our case it was a “weekly scrum” instead. According to scrum we first tried to schedule the meeting at the same time and place every week, but later we had to limit ourselves to the same weekday and building. This was due to our different schedules and geographical problems as well as the difficulty in booking the same group room every week. We chose times where all team members were able to attend and postponed the meeting if someone was missing. By doing that we disregarded some scrum-principles but we valued each group member´s participation.

The weekly meeting was intended to be a communication channel for the group and worked as a status update of the achieved sprint tasks. It also enabled follow-up conversations, as well as identifying and solving our current programming and documentation issues.

Single Programming

Overall the group chose to programme separately because of the major risk for a teacher-student relationship and the "*Watch the Master*" phenomenon which can arise if one member is much more experienced than the other. We wanted to avoid situations where the less experienced member took an observer role, while the more experienced member stands for majority of the coding due to the fact that this leads to disengagement by the one watching.

By doing this we missed out on the benefits that comes of pair programming as better code quality and spreading skills throughout the group. It also led to a longer time before we could solve possible problems since the group only met once a week.

Pair Programming

When facing difficulties on our own we used pair programming after the weekly meeting to solve the problems. This was to allow another member of the group to examine your code, give feedback and provide help with unsolved issues. It almost always led to a teacher-student situation where the one helping told the other one how to correct the code.

Because the usage of this programming principle in the beginning only was used when problems already had surfaced and not as a way of building our application, we did not reach the principle's full potential. Due to the new group and the lack of time we could not expect for the group members to be good at it from the start, or even have the time to learn to be good at it. Pair programming takes too much time while considering the limited amount of time available to the group.

Towards the end of the project Skype was used to enable pair and group programming, even though the group was spread out over large geographical areas. This made it possible to gain some of the benefits of pair programming, despite the fact that we were not actually sitting next to eachother.

Test Driven Development

We used TDD when building the application's user interface, which means that after implementing a new method or function we tested it to ensure the quality and functionality of the application. For the other classes tests were made when needed, which was mostly when the class was first made or upgraded. This was because writing and running tests costs time, and therefore TDD seemed too slow of an approach for this short project. Instead, we tried to thoroughly document the code during the whole project, so that possible flaws could be more easily discovered afterwards.

Object Oriented Programming and Model-View-Controller

The programming method chosen was Object Oriented Programming using the design structure MVC with helper classes. This has given the project a structure which separates between different classes, views and user interfaces. As a result, it was possible to rearrange and rebuild classes without the whole application crashing. One clear example of this is the local class containing the data of the alarms, which has been redesigned a lot. It was first built with lists, then we developed helper classes and finally the databases. The functionality of the application was maintained through each redesign phase, as well as when extensions were added, because of the separated structure and the proper level of abstraction.

Distribution of time

7,5 hp equals 200 hours which was divided between lectures, tutoring sessions, group meetings and the development of the application either on our own or with the whole group. The percentage of time spent on each development task by each group member is shown in the table.

Time distribution	Programming	Testing	Documenting	Managing technical aids
Alexander	70%	5%	10%	15%
Sebastian	80%	5%	15%	0%
Isabel	20%	10%	60%	10%
Joakim	65%	10%	5%	20%
Gabriella	30%	10%	50%	10%
Emma	30%	0%	60%	10%

Reflections over non process specific decisions

This section presents the different elements that have influenced the work process, and therefore the development of the application. It describes the difficulties and opportunities that were present during the project.

New group with different backgrounds

Difficulty - We were a group of people from different study programmes whom did not know each other beforehand. We all had different experiences and preferred different ways of working as a team, for example at what time we should start working in the morning. The guys were and are typical "IT-guys" who wanted to make their own decisions and just get things done without much collaboration and communication. The girls wanted the opposite, that everyone should be *"on the same page"* in the process and wished for no major decision to be made separately without the rest of the group knowing. They were probably more used to working in larger groups and cooperating, while the guys have worked more in smaller groups of two and two, where communication is not as important as in larger groups.

Opportunity - We got to work in a team with no ingrained opinions. We started off in a really high speed and at first we tried to make everyone invested in the process and the programming. Everyone wrote code to their own ability. As the requested level of expertise got higher and the programming got harder and more difficult, the group had to make new arrangements. Some of us did the programming and the others did the documentation, which was how we all expected it would eventually turn out, but we tried to postpone it as long as possible. The positive side of our varying backgrounds was that we had expertise in both programming, documentation and group collaboration, which in all areas help to raise the quality of the application's technical aspects and the documentation supporting it.

Holding the group together

Difficulty - The group has had some difficulties with arriving on time for meetings, which is only one furthermore reason why this application is a great idea. The time passing has improved over time, and at least at the end of the project nearly all of us communicated with the others if one was running late. If our application had been finished sooner maybe we could have eliminated this problem completely.

Opportunity - Although, or maybe because of, we had never met before we managed to have an atmosphere that were both nice and helpful. We dared to ask all the silly questions and to communicate very openly. Every member of the group were contributing and we made sure that everyone always got the same information. All the way through we have had

respect for each others differences and we have all gotten our time to shine, even only if for a brief moment.

We all trusted one another to do our best and to make the right decisions. Some of that trust, however, could have been a result of lack of knowledge about how to solve a specific issue. As a result, group members might have felt that they did not have the basis to question the others, and therefore left the decision making in the hands of the ones responsible for each task.

One reason behind the positive feeling in the group might have been the weekly “fika” during the meetings. Who does not get happy when having a little fika?

Different schedules and locations

Difficulty - It was a bit of a puzzle to find time to work together and to get all the different schedules in order. Due to the fact that one person was living in Oxford, two others were writing their bachelor thesis and all had other courses and activities to attend. We tried our best and eventually found a system which worked for everyone in the group. This was the reason to why the group only had weekly meetings and had to work separately most of the time.

Opportunity - We focused on keeping a good atmosphere the few times we met. The time time was too valuable for focusing on group members properties that might have become an issue for the group feeling in the long run. Now we accepted each others personalities as they were only focusing on the positive.

Meetings and working together on wednesdays only

Difficulty - In the beginning we had some trouble with the structure of the meetings, but as we got to know each other better the meetings got easier and clearer. The group sometimes had follow-ups on Facebook and Skype during the weeks. This was to coordination meetings or to discuss technical issues. Due to the fact that some of us were able to figure out a solution on their own when they got stuck and others did not, meeting only on wednesdays was too little time together. We could have gotten a lot more work done if a problem that was easily solved by one group member did not result in another member getting stuck for the entire week.

This also led to no existing coding style or standard. Every group member wrote the code their own way and chose solutions known to them and unknown for others. This does not always lead up to the best possible solutions. Discussions among group members could have gained different perspectives and improvements of the code.

Opportunity - While we had so little time together we really learned to work disciplined and in a structured manner those days we had together. It was also a good incentive to keep the deadlines for the sprints so that we could use the time together to solve possible problems and take new major technical decisions for the group's progress. Maybe more time together would have led to more ineffective time and not the same level of focus.

Unequal level of knowledge in programming

Difficulty - The unequal programming ability led to an uneven result in programmed rows though the group committed the same amount of time. It also led to unequal decisionmaking due to the fact that some group members did not have enough insight. Because of this we might have missed out on some influences to the application and our work that would have been valuable.

Opportunity - The ones less skilled in programming hopefully got to learn a little bit more about it through the pair programming sessions we had, even though it worked more as mentoring sessions than actual pair programming. The ones more skilled got to train their patience and ability to teach. This might have been a little bit frustrating for some though, but it made us all a tiny bit better people.

Advanced application

Difficulty - The application might have been too advanced for the way we chose to work and could probably have been finished much earlier if we worked more together and more frequently. It was way to advanced for the girls in the group with less experience in programming, which led to only half the group programming in the end. Due to the late completion of the application, the documentation also got harder and a bit stressful. For example, it is hard to write a user manual or an uml-diagram to an incomplete application that may change. Perhaps the application also was too advanced for the amount of time spent on this course because of the fact that so much time went to research instead of programming. The development of the basic alarm took too much time and postponed the completion of the most crucial feature, which is the network activity.

Opportunity - For those who programmed all the way it must have been an exciting project which forced them to learn a lot of new things. Hopefully it will help in future projects and application developments. The others really learned to be flexible and to work under pressure when both programming and writing the documentation.

What we would have done differently in a future but similar project

When working on a project one can always find both improvements and changes if given the chance to start over. If we would have done this over again in the same group constellation we would have done some things differently.

The applications level of difficulty

Our group was put together during the first week of the project because we were all without a group from the beginning. Therefore some of us became group members when the idea for the application was already chosen. This entails that the idea for the application was chosen before we had full knowledge about the whole group's competence in java and Android. This became a problem since the project was very complicated for some of us, hence others had to take a greater responsibility in the programming to enable us to reach our vision. If given the chance to do the project again we would have chosen to make an application that would have been more technically achievable for all members of the group.

Due to this knowledge gap, the technical decisions made resulted in longer programming time than was anticipated during the decision process. If we would have acknowledged all group members differences in experience we could have chosen more achievable goals where all group members could have taken a greater part i the programming process.

More frequent meetings and work sessions

We have had a good connection between the group members, even though we have only met all together on average one day per week, and few of us knew each other beforehand. Given the possibility to work together again, we realise we would have been able to improve the application if we could have met more often. During this project we sometimes took a fika together and we also had plans to eat dinner together as a bonding activity. Sadly the dinner did not occur but it is something we would make sure of doing if given the opportunity to work together again. We believe that a good connection between the group members is an important key factor to establishing higher quality in the project as a whole, and that it makes the process more fun.

Documenting all code continuously and directly

A lot of the code documentation was thought of retrospectively. This made it difficult to correct and look over other group members code. This really taught us the importance of continual documentation, and is something we would be stricter on if doing this project a second time around.

Not only the documentation of code was retrospective, but also the overall documentation was sometimes lacking behind. Although we continuously wrote the sprint backlog and had written all the user stories at our first ever meeting together, other documents like the design decisions, acceptance test and others, were written a while after they had already been done. In the next project this would be something that we would have written the moment while the decision or test was made, to make sure no valuable information was missed and to save time.

Getting the whole group involved in the decision process

Because of our knowledge gap within the group a lot of the technical decisions were made by those with more technical experience. As a result, a lot of the technical decisions made could not be fulfilled by all members of the group. The complexity of the application became a hurdle since a lot of the code produced by those not familiar with the complex attributes of the application could not be fully used, and sometimes had to be rewritten by those with more knowledge of programming. Doing a similar project again this would be a hard decision for our group since all group members want to develop their skills from their own knowledge level, and finding an application idea to suit all members possibilities would be something worth thinking about carefully.

Make the most of the competences in the group

As being said, those with less programming experience has a lot of experience in documenting and presenting as well as many ideas for new products with market value. In this perspective those with this competence has been of great value for structure of meetings and documentation of this project. Due to the circumstances we had in this particular project the delegation of tasks has been efficient for our projects success. The right person at the right place. In a future project we would most likely try harder for all group members to be able to leave their comfort zone and have a more equal distribution of different tasks. It is easy to fall back on only programming if you do not like writing documentation, likewise it is hard for someone with little java experience to program an Android database in the same amount of time as someone with experience. Even though all of us has put down a lot of time trying to understand and to develop the code not all of us

were able to contribute as much as others to the features of the application. To avoid this in a future project the technical decisions should be more suited for the whole group's knowledge base.

Trying not to "down prioritize" this course due to other heavy courses

A great factor to why we had to distribute the tasks to the person with most knowledge on the specific subject, instead of spending more time learning something new, was due to our other courses running parallel to this one. This period we all had time consuming courses aside this one, and that diminished the extra time needed for all of us to learn more about new areas. If we would have done this again, given we had chosen a simpler application idea and did not have the geographical problems that we have had this time, we would work more together both with pair programming as well as the documentation and presentation. To become a successful software developer one needs to be able to see new and attractive solutions and the ability to successfully pitch these to an investor, as well as the ability to develop the application itself. Therefore, the competence of each of the group's members is relevant for this project, however, given the chance of doing a similar project again in the future we would try to learn from each other to a larger extent.