Luleå University of Technology

Third year project

Sensor data aggregation through CoAP

***A*uthors:**
Sophia Bergendahl
sopber-8@student.ltu.se

Edvin Bruun
edvbru-9@student.ltu.se

William Gustafsson
wilgus-9@student.ltu.se

Christoffer Holmstedt
cihhol-7@student.ltu.se

Marcus Rådman
marrdm-9@student.ltu.se

Kristoffer Svensson
kirsev-9@student.ltu.se

Ludwig Thurfjell
ludthu-7@student.ltu.se

***S*upervisors:**
Ulf Bodin
ulf.bodin@ltu.se

Jens Eliasson
jens.eliasson@ltu.se

Rumen Kyusakov
rumen.kyusakov@ltu.se

March 4, 2012

Project Description

Background

Luleå University of Technology conducts research on lowpower wireless microprocessors called "Mulle". These microprocessors can be used for various things depending on which type of sensors you connect to it, everything from measuring temperature or vibrations in a car to analyzing the quality of the road that you drive on.

Every year northern parts of Sweden are used for testing cars during winter conditions. To test a car you first decide what you want to test, then you test with local sensors logging within the car. When enough data is collected you return back home. At the testing facility the data is now available for analysis. Depending on the results from the previous runs you might want to test some parts in more detail so you re-configure all sensors and go out for another test run.

This process is time consuming when you need to return to testing facility to be able to analyze and re-configure all sensors. In todays society most computers are connected to internet and/or other private networks, most of these computers have the ability to be remotely configured and maintained. The goal with this project is to be able to analyze data from sensors in realtime and re-configure them on the fly while testing is in progress.

Project Targets

1. Be able to send live sensor data from multiple "Mulles" to an online logging server/service.

2. Be able to read sensor data on the web with both a PC (web browser) and through an Android mobile device.

3. Be able to re-configure the sensors through a web interface and through an Android mobile device.

Technical dilimiations

TODO: Vad har explicit uteslutits från arbetet?

Execution of the project

Scrum and how it has been used

It was decided back in november that the entire project would be divded into three sprints. The exact dates were to be decided in the beginning of each sprint. In cooperation with the client the scope of the project and the scope of the first sprint was decided upon in november. During the first projectmeeting the first sprint goal was divided into eight sprint stories. It soon became clear that those eight stories were way to big, at the end of the sprint none of the stories had been finished.

Lesson learnt, the second sprint was divided into smaller stories which gave immidiate result when the first 69 sprint story points finished during the second sprint.

To decide upon size for each sprint story, for the second and third sprint, "planning poker" [1, p. 42] was used. For every sprint story each project member wrote down an estimate on the scope for each story. With planning poker it became clear that each project member had a different vision for each story. A short discussion after each estimate made it more clear on how big the scope was, an agreement was usually made within a few minutes.

One project, three sprint goals

As mentioned earlier the project was divided into three sprints. This meant that three different sprint goals had to be divided into smaller sprint stories which in turn had to be assigned to a project member. Due to all project members being new to most of the tasks at hand the first team division was made with focus on components [1, p. 106]. The goal with this was that each smaller team within the project could sit together and dive deep into their specific part such as the Mulle, the server parts or the android code. Later on a split into cross-component teams [1, p. 107] was aimed for but due to some persistent bottlenecks in some components this was never done.

For all sprints the sprint planning meeting were used to categorize each sprint story into the different components (Mulle, Server, Android). It was then up to each component based team to split their stories between themselves. This ended being a very flexible solution, in some cases, to flexible when a team didn't use the Scrumboard online at Scrumdo.com the team could wander off from the sprint story they were supposed to work on. For the last sprint everyone got at least one sprint story assigned to themselves directly during the sprint planning meeting. This was made to put more focus on using the Scrumboard at Scrumdo.com.

A move to cross-component teams was never made instead an attempt to increase speed for the Mulle and the Android component was made by moving one team member from the server team to the Mulle team and another to the Android team. The server component at this time was way ahead of the other components. Another week later it became clear that the additional team member for Mulle team was not needed, cause the problem was still a bottleneck that only one or two team members could work on at a single point in time, a move back to the server team was made.

Individual time monitoring and speed

Sophia Bergendahl

Edvin Bruun

William Gustafsson

Christoffer Holmstedt

Exempelvis kan lista användas om man vill. Jag, christoffer, kommer inte använda det tror jag.

1. TODO: Hur man citerar till specifik sida i kurslitteraturen. [1, p. 42]
2. TODO: Hur man citerar utan sidhänvisning [1]
3. Third sprint story

Marcus Rådman

Kristoffer Svensson

Ludwig Thurfjell

Reflection about Scrum usage during this project

Alot has been learnt during this project and to keep it short the following lessons learnt and improvements that can be made are a chosen few.

At the end of all projects when the deadline is closing in the pace of the development often increases. A downfall of this is that when the speed increases the quality of the code often decreases. Scrum is a solution to this problem with the main goal of keeping a steady development pace and always keeping the quality of the code as good as possible above some minimum criteria. With to long sprints, projects will still end up with a big deadline, this is what happened in this project. In total the project lasted about 17

weeks including the winter holidays. Instead of three sprints where each lasted about five weeks a project split into smaller sprint would have been better. If time travel was possible this project would have had two smaller sprints in the beginning each would have lasted for two weeks. The first sprint with goal of configuring all required software for everyone such as setting up git, the different IDEs, gcc and other required software/tools. The second sprint with the goal of understanding what was available at the time and get all basic functionality working. If the project had found any big bottlenecks at the end of the second sprint that would be the point in time to halt the project and really rethink what to focus on. The remaining weeks should have been divided into three evenly sized sprints.

Another big improvement could have been made in the relation between Scrum team and product owner. From the beginning it was unclear who was the product owner out of three supervisors/clients, this should have been defined to one single person as early as possible before continuing with any other work. The absence of the product owner before and during our sprint planning meetings resulted in alot of confusion when it was up to the team to decide an estimate for each sprint story. The lesson learnt from this is that without a product owner the scrum team will fumble in blindness forever or as Kniberg writes [1, p. 25] *"...each story contains three variables that are highly dependent on each other"*. There is no way the team can choose a good estimate when there is no product owner that has already chosen an importance and scope for each sprint story to start with and is available to change that during the course of a sprint planning meeting.

The last and perhaps the most important improvement that will be mentioned is the importance of having a team member taking the role as Scrum master. Without the Scrum master during the first sprint everyone was on their own. For the second and third sprint the Scrum master role was appointed to one team member. This improved the communication within the Scrum team alot and also made it possible for individual team members to have someone to go to in case of general questions and/or other problems.

TODO: ...samt redovisning av arbete med krav mot kravställare och hantering av projektvariablerna omfattning och kvalité.

Results

Deliverables

Mainly due to being unable to setup communication from the Mulle to the server, project target number one listed in the beginning of this document is not met. What is delivered from the project concerning this is available in Appendix A where a guide on how to get started where this project ends is available.

Project target number two is not met. What is delivered from the project is a basic webapplication with a simple database structure as the back-end where future real sensor data can be stored.

Project target number three is not met. Basic functionality on how to use EXI as encoding for sensor data is available for the Mulle and the Android application but hasn't been tested. No work on EXI has been made for the server though there exist a simple webpage to turn a value on or off.

Testing

During this project testing has been made by each project member. The scope of the testing for each story has been up to each project member to decide upon. Right before each sprint demo a project meeting was scheduled were each one showed what was completed and what was not. Depending on what was ready at that point in time the entire Scrum team decided what was going to be presented at the demo and a test was made to confirm that it was possible to show the parts that were decided upon. No further testing was made during this project.

Lessons learnt

Communication is always troublesome, it's usually easy to get a system to transmit data and another to do the same. The hard part is when you want different system to actually communicate with each other. This boiles down to the lesson learnt that the project should have focused more on specific components instead of spreading out on all three components (Mulle, Android and the server) from the beginning. The goal should have been to get the Mulle communicate with either the server or the android device to start with. When that was operational new systems could have been added to the mix.

Testing is always hard to do, it might be easy to test specific test cases but is very hard to test all possible usages. To make sure that your codebase doesn't become useless in the long run you need somekind of automated

testing to make sure that all previous bugs found is tested automatically with all new improvements and add-ons. It might be time consuming in the beginning of a project but it's priceless in the end. An important part of testing is to test both individual components and interaction between different components. If only testing is made to the interaction between different components a finished component cannot be tested until the other one is finished aswell, this creates bottlenecks for the entire project.

When is something done? Do not let this question be up to the individual programmer, it puts the programmer in a difficult situation. Either the programmer wants to create the perfect spot and keeps going forever or the programmer will take shortcuts that will comeback and haunt you later on. It must be up to the team to decide when something is done or not. This will make it alot easier in the day to day work for everyone, if anybody is uncertain if it's completed or not, just look into what was decided earlier.

Suggested improvements

Without meeting any of our project targets it's hard to suggest improvements except the obvious one to complete what has been started. Instead of going straight at it and try to finish it all as soon as possible some thought process need to be put into the question, what really needs to be finished?

From this projects point of view the first thing to prioritze is to get some communication with the Mulle. The Mulle in its current state has alot of bugs which makes it hard to debug, there seems to be alot of "random" bugs occuring when you lest expect them. The code that is finished "should work" but hasn't.

The second priority should be to rethink the servers purpose. At the current state the python implementation is very simple and if the only thing you want to do is add new CoAP services which just return some simple results, then keep going with the current server implementation. A new service is very easy to add. What has been realized at the end of this project is that in the long run it might be better to create a new C server implementation instead. The main reason for this is that when you want to implement custom communication protocols such as EXI alot of code is available in C and with a server implementation in C it will be relatively easy to share code between the server and Mulle system.

As the third and last priority, work should be put into the Android system. With the Mulle up and running for testing purposes it should be easy to get going with the android application but without the Mulle the android application isn't worth anything.

Conclusions

References

[1] Henrik Kniberg, *Scrum and XP from the Trenches*. C4Media Inc, Publisher of InfoQ.com, 978-1-4303-2264-1, http://infoq.com/minibooks/scrum-xp-from-the-trenches, 2007.

Appendix A - How to build upon our codebase

This appendix include information on how to build upon our codebase for the Mulle (C), server code (Python, PHP/HTML5 and C) and Android Mobile phone (Java).

Mulle

Server

Coapy server

TODO: Python parts such as the python coapy server and how we use EXIP c-code parts.

Webpages and database

Android Mobile Phone application