10. June 2016 by CodeLionX on Allgemein

HW20: Installation

Dear Blog-Readers,

today we would like you to take action. We have build our Android App as an APK and published it on Github.

Please follow this guide to install our Android App on your device. We would really appreciate if you can note your actions and results. You can use the following table for that:

Executor	name		
Date	date		
Device Model	device name	**Android Version**	version
Signup successful:			X
Comments:			
Security Settings Change successful:			
Comments:			
Download successful:			
Comments:			
Application start successful:			
Comments:			

Login successful:
Comments:
Tested Application successfully:
Comments:

Please submit the table to us via our comment functionality or via email to unveiled@gmx.de, so that we can analyse the test result.

Regards

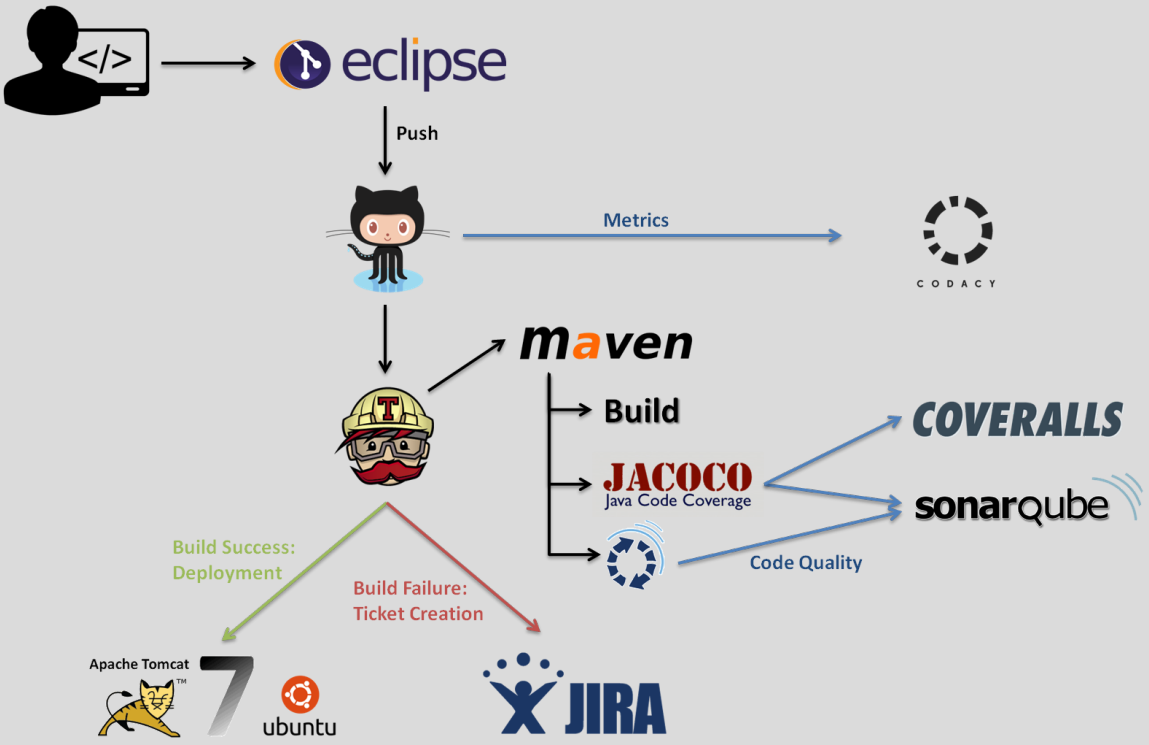
CodeLionX

30. May 2016 by CodeLionX on Allgemein

HW19: Automatic Deployment

Hi together,

today we want to describe you our deployment process. We used TravisCI for Continuous Integration and you can find our build of our streaming library imflux here and for our Java-Backend-Stack here. The following picture shows our build lifecycle and the tool chain we use for our Continuous Integration (Continuous Delivery).

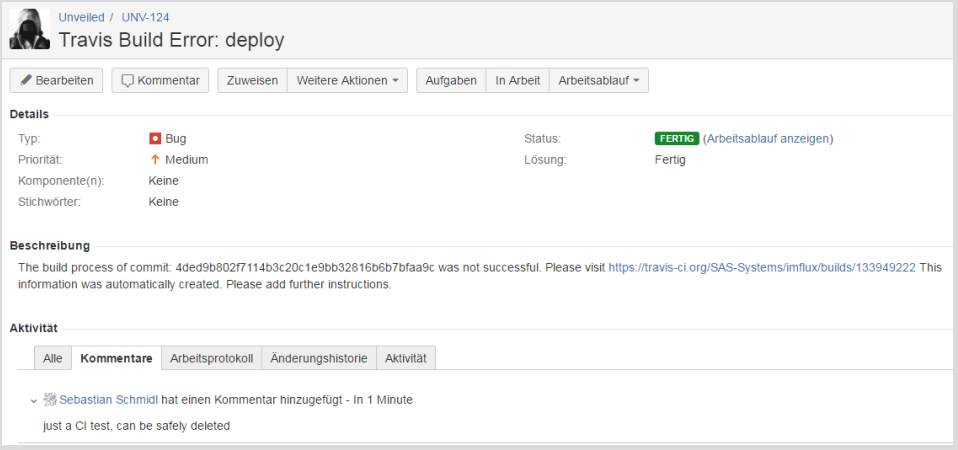


We use the following tool chain (for the Unveiled-Server build):

- `mvn install -DskipTests=true -Dmaven.javadoc.skip=true -B -V`
  We use Maven for dependency management and building our application.
- `mvn clean test jacoco:report`
  JaCoCo is used for running JUnit tests and collecting the test results.
- If the build was successful following things are done:
  - `mvn coveralls:report`
    The test results are reported to coveralls.io to make them available for analysis (currently there are no test for the Unveiled-Server build, but you can check the imflux test results).

- `mvn sonar:sonar -Dsonar.host.url=$SONAR_HOST_URL`
  Sonarqube analyses the binary files and reports the results to the website.
- `mvn tomcat7:deploy -Ddeploy.url=$DEPLOY_URL -Ddeploy.username=$DEPLOY_USER -Ddeploy.password=$DEPLOY_PASSWORD`
  The builded .war file is deployed to our own server using the tomcat7 maven plugin.

- `./send_jira_ticket.sh`
  If the build was not successful a new Jira ticket is created and sent to the Jira server. Therefore we use a short shell script using curl to send a Json to the Jira API. You can see the result in the following picture:



All this information can be found in our travis.yml of the repositories imflux and Unveiled-Server.

Have a nice week

CodeLionX

30. May 2016 by CodeLionX on Allgemein

HW18: Metrics

Hey folks,

today we want to share with you, how code metrics helped us to improve our code quality. We have done the metrics analysis within our eclipse IDE and the tool Metrics 1.3.6. Metrics 1.3.6 allows us to run low level analysis. Unfortunately we were not able to find a comparable tool which allows this low level analysis within our build lifecycle. But you can take a look at our Sonarqube projects (imflux, Unveiled-Server). Sonarqube shows some high level metrics.

The following picture shows the results of Metrics 1.3.6 of the Unveiled-Server project before our changes.

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1,02	1,769	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
Number of Static Attributes (avg/max per type	30	2,308	1,488	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,154	0,411	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragr	13	4,333	1,247	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	50	3,846	4,671	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,154	0,361	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1033					
Weighted methods per Class (avg/max per typ	162	12,462	10,66	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	97	7,462	8,828	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Depth of Inheritance Tree (avg/max per type)		1,308	0,722	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,636	1,743	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
java		1,636	1,743	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
sas.systems.unveiled.server.fileio		1,533	2,115	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet.java		3,667	5,121	15	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileWriter.java		2,333	0,745	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	writeToFile
FilePOJO.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
sas.systems.unveiled.server		1,8	1,579	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server.util		1,625	0,992	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	SessionManager
resources		0	0			
Nested Block Depth (avg/max per method)		1,323	0,664	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,383	0,396	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	572	5,778	10,64	84	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
Number of Overridden Methods (avg/max per	2	0,154	0,361	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragn		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

As you can see we have two possible points where we are not in the green (in this case blue) area. The first one is the number of parameters, where we have a maximum of 16 parameters per method. This method is a constructor of our File POJO class. We decided to not change this class, because it represents a database record. Instead we decided to change the Cyclomatic Complexity of our `doPost()` method our `FileUploadServlet` class. You can find the code before the refactoring in our Github-repository.

To improve our code quality and the cyclomatic complexity of the `doPost()` method, we extracted some logic into new methods. Therefore it was necessary to create a new class: `FileParameters` and a new exception class: `BadRequestException:`

```
        /**
         * Exception class for the parameter parsing meth
```

```
         *
         * @author CodeLionX
         */
        private class BadRequestException extends Excepti

                private static final long serialVersionUI

                public BadRequestException(String string)
                        super(string);
                }
        }
```

You can find the new source code here. The following screenshot shows you the metrics analysis result after refactoring:

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		1,019	1,716	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
Number of Static Attributes (avg/max per type	31	2,067	1,526	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,118	0,375	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragr	15	5	0,816	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	51	3,4	4,499	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1053					
Weighted methods per Class (avg/max per typ	167	11,133	10,893	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	104	6,933	8,575	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Depth of Inheritance Tree (avg/max per type)		1,4	0,8	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,575	1,181	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
java		1,575	1,181	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server		1,8	1,579	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
sas.systems.unveiled.server.fileio		1,423	0,948	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet.java		2,167	1,462	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileUploadServlet		2,273	1,483	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
doPost	6					
readRequest	3					
authenticateUserWithToken	3					
getDoubleSilently	3					
getIntSilently	3					
getBooleanSilently	2					
FileUploadServlet	1					
init	1					
destroy	1					
createDbEntry	1					
writeFile	1					
BadRequestException		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	BadRequestException
FileWriter.java		2,333	0,745	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	writeToFile
FileParameters.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileParameters
FilePOJO.java		1	0	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
sas.systems.unveiled.server.util		1,625	0,992	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	SessionManager
resources		0	0			
Nested Block Depth (avg/max per method)		1,34	0,671	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,337	0,395	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	596	5,623	7,717	37	/Unveiled-Server/src/main/java/sas/systems/unveile...	insertFile
Number of Overridden Methods (avg/max per	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragn		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

So you can see that we now have reduced the cyclomatic complexity of the `doPost()` method from 15 to 6 by introducing six new methods and two new classes.

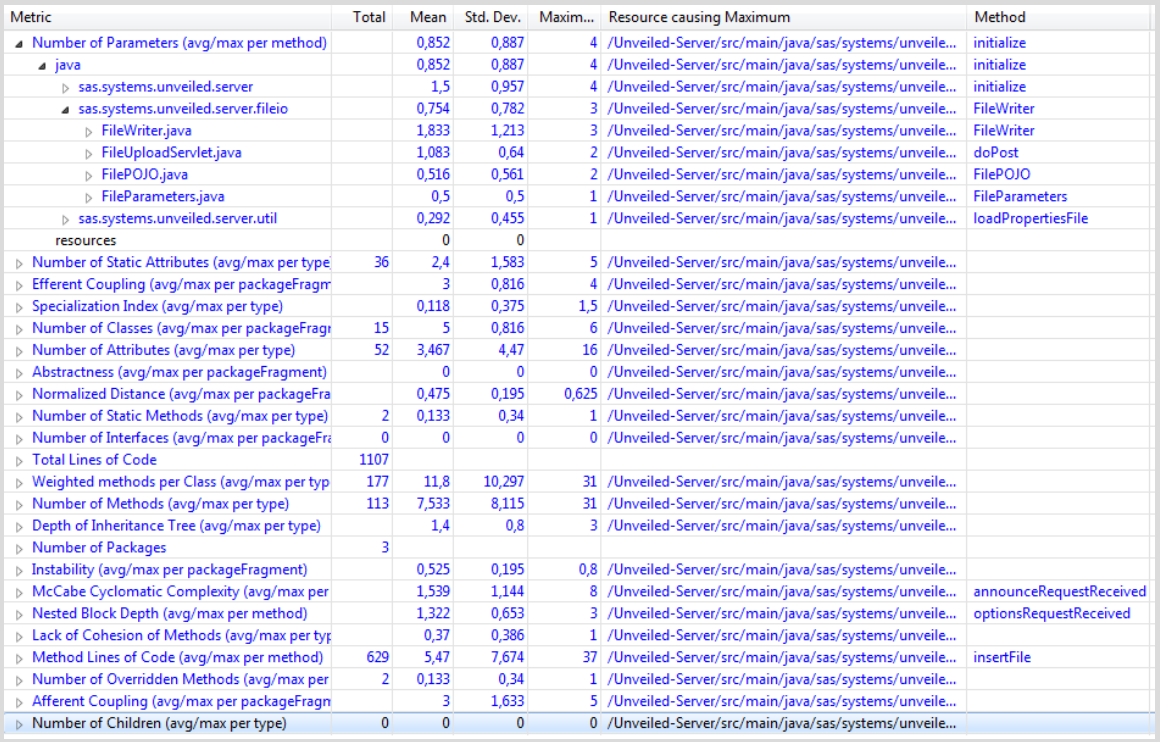
EDIT:

As it turned out that we had to optimize two different aspects, we have done that later on:
As we only have 2 metrics that are not good, we have also optimized the one we stated before as not necessary. This is the Metric *Number of Parameters*, which is bad for our `FilePOJO` java class. See the screenshots below:

Screenshot before the second refactoring:

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0,974	1,646	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
java		0,974	1,646	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
sas.systems.unveiled.server.fileio		0,984	2,051	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
FilePOJO.java		0,97	2,702	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
FileWriter.java		1,833	1,213	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileWriter
FileUploadServlet.java		1,083	0,64	2	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FileParameters.java		0,5	0,5	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileParameters
sas.systems.unveiled.server		1,5	0,957	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	initialize
sas.systems.unveiled.server.util		0,292	0,455	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	loadPropertiesFile
resources		0	0			
Number of Static Attributes (avg/max per type	36	2,4	1,583	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,118	0,375	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragr	15	5	0,816	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	52	3,467	4,47	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1106					
Weighted methods per Class (avg/max per typ	179	11,933	10,554	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	115	7,667	8,506	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
java	115	7,667	8,506	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
sas.systems.unveiled.server.fileio	63	12,6	10,929	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
FilePOJO.java	33	33	0	33	/Unveiled-Server/src/main/java/sas/systems/unveile...	
FileParameters.java	12	12	0	12	/Unveiled-Server/src/main/java/sas/systems/unveile...	
FileUploadServlet.java	12	6	5	11	/Unveiled-Server/src/main/java/sas/systems/unveile...	
FileWriter.java	6	6	0	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
sas.systems.unveiled.server	29	7,25	5,932	17	/Unveiled-Server/src/main/java/sas/systems/unveile...	
sas.systems.unveiled.server.util	23	3,833	4,776	14	/Unveiled-Server/src/main/java/sas/systems/unveile...	
resources	0	0	0			
Depth of Inheritance Tree (avg/max per type)		1,4	0,8	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,53	1,137	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
Nested Block Depth (avg/max per method)		1,316	0,649	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,37	0,386	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	620	5,299	7,454	37	/Unveiled-Server/src/main/java/sas/systems/unveile...	insertFile
Number of Overridden Methods (avg/max per	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragm		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

I changed the constructor to only get two parameters and set the other attributes to default values. You can see all the changes here.
And this is the result of the metrics tool after the second refactoring:



Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0,852	0,887	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	initialize
java		0,852	0,887	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	initialize
sas.systems.unveiled.server		1,5	0,957	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	initialize
sas.systems.unveiled.server.fileio		0,754	0,782	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileWriter
FileWriter.java		1,833	1,213	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileWriter
FileUploadServlet.java		1,083	0,64	2	/Unveiled-Server/src/main/java/sas/systems/unveile...	doPost
FilePOJO.java		0,516	0,561	2	/Unveiled-Server/src/main/java/sas/systems/unveile...	FilePOJO
FileParameters.java		0,5	0,5	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	FileParameters
sas.systems.unveiled.server.util		0,292	0,455	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	loadPropertiesFile
resources		0	0			
Number of Static Attributes (avg/max per type	36	2,4	1,583	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Efferent Coupling (avg/max per packageFragm		3	0,816	4	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Specialization Index (avg/max per type)		0,118	0,375	1,5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Classes (avg/max per packageFragr	15	5	0,816	6	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Attributes (avg/max per type)	52	3,467	4,47	16	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Abstractness (avg/max per packageFragment)		0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Normalized Distance (avg/max per packageFra		0,475	0,195	0,625	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Static Methods (avg/max per type)	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Interfaces (avg/max per packageFra	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Total Lines of Code	1107					
Weighted methods per Class (avg/max per typ	177	11,8	10,297	31	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Methods (avg/max per type)	113	7,533	8,115	31	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Depth of Inheritance Tree (avg/max per type)		1,4	0,8	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Packages	3					
Instability (avg/max per packageFragment)		0,525	0,195	0,8	/Unveiled-Server/src/main/java/sas/systems/unveile...	
McCabe Cyclomatic Complexity (avg/max per		1,539	1,144	8	/Unveiled-Server/src/main/java/sas/systems/unveile...	announceRequestReceived
Nested Block Depth (avg/max per method)		1,322	0,653	3	/Unveiled-Server/src/main/java/sas/systems/unveile...	optionsRequestReceived
Lack of Cohesion of Methods (avg/max per typ		0,37	0,386	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Method Lines of Code (avg/max per method)	629	5,47	7,674	37	/Unveiled-Server/src/main/java/sas/systems/unveile...	insertFile
Number of Overridden Methods (avg/max per	2	0,133	0,34	1	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Afferent Coupling (avg/max per packageFragn		3	1,633	5	/Unveiled-Server/src/main/java/sas/systems/unveile...	
Number of Children (avg/max per type)	0	0	0	0	/Unveiled-Server/src/main/java/sas/systems/unveile...	

As you can see, we improved the maximum of the metric *Number of Parameters* from 16 to 4.

Have a nice week

CodeLionX

o

22. May 2016 by CodeLionX on Allgemein

HW17: Test plan and coverage

Hi together,

today we want to provide you the link to our testplan. This document describes all the test we are doing to ensure we

have no bugs in our new implemented features. You can find it in under documentation.

Additionally we also reached our goal to have more than 50% test coverage for our streaming library imflux. You can find all test reports on coveralls or on sonarqube (Please be aware that you might not be able to use this link inside the DHBW network. Please use this link instead.).

Have a nice week

CodeLionX

8. May 2016 by CodeLionX on Allgemein

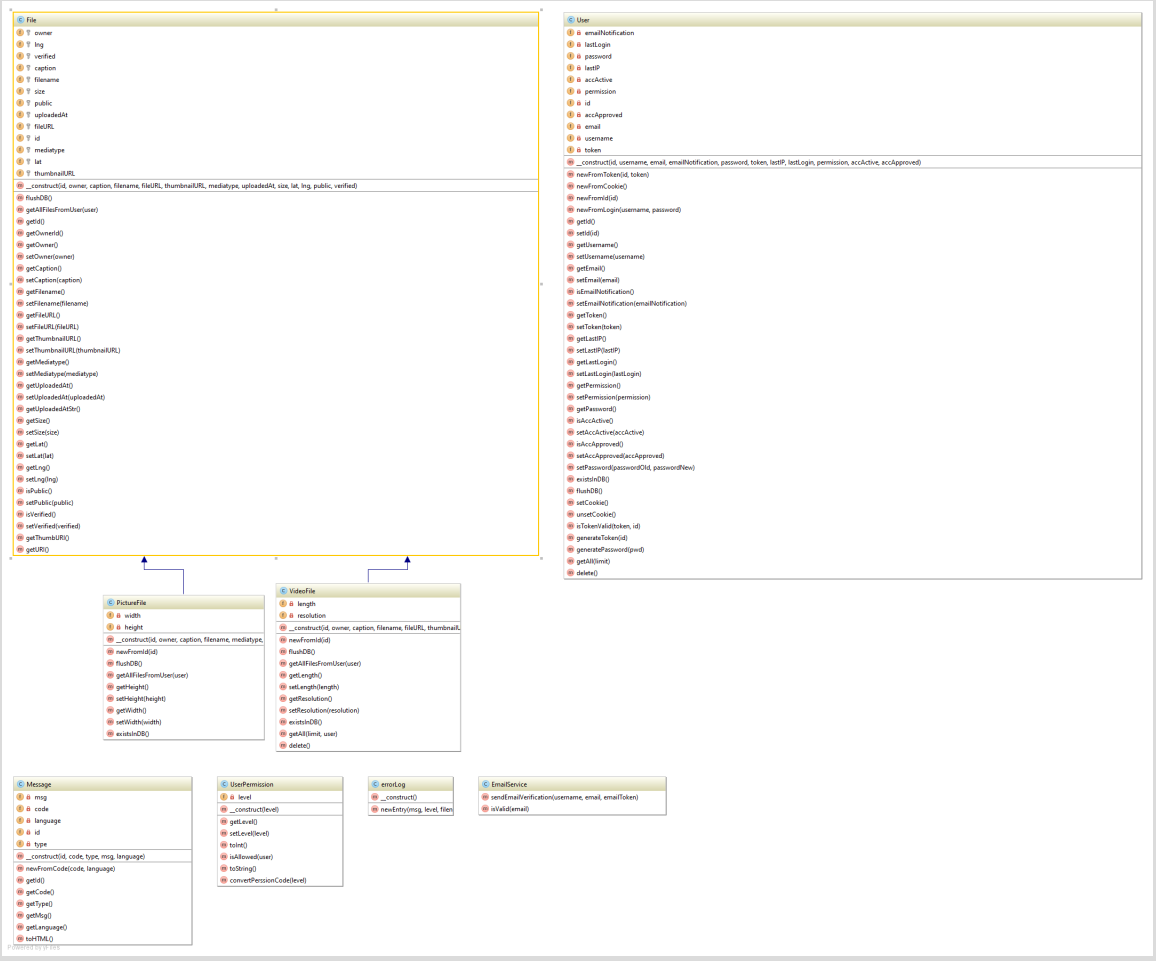
HW16: Implement Design Pattern in Project Code

Hi together,

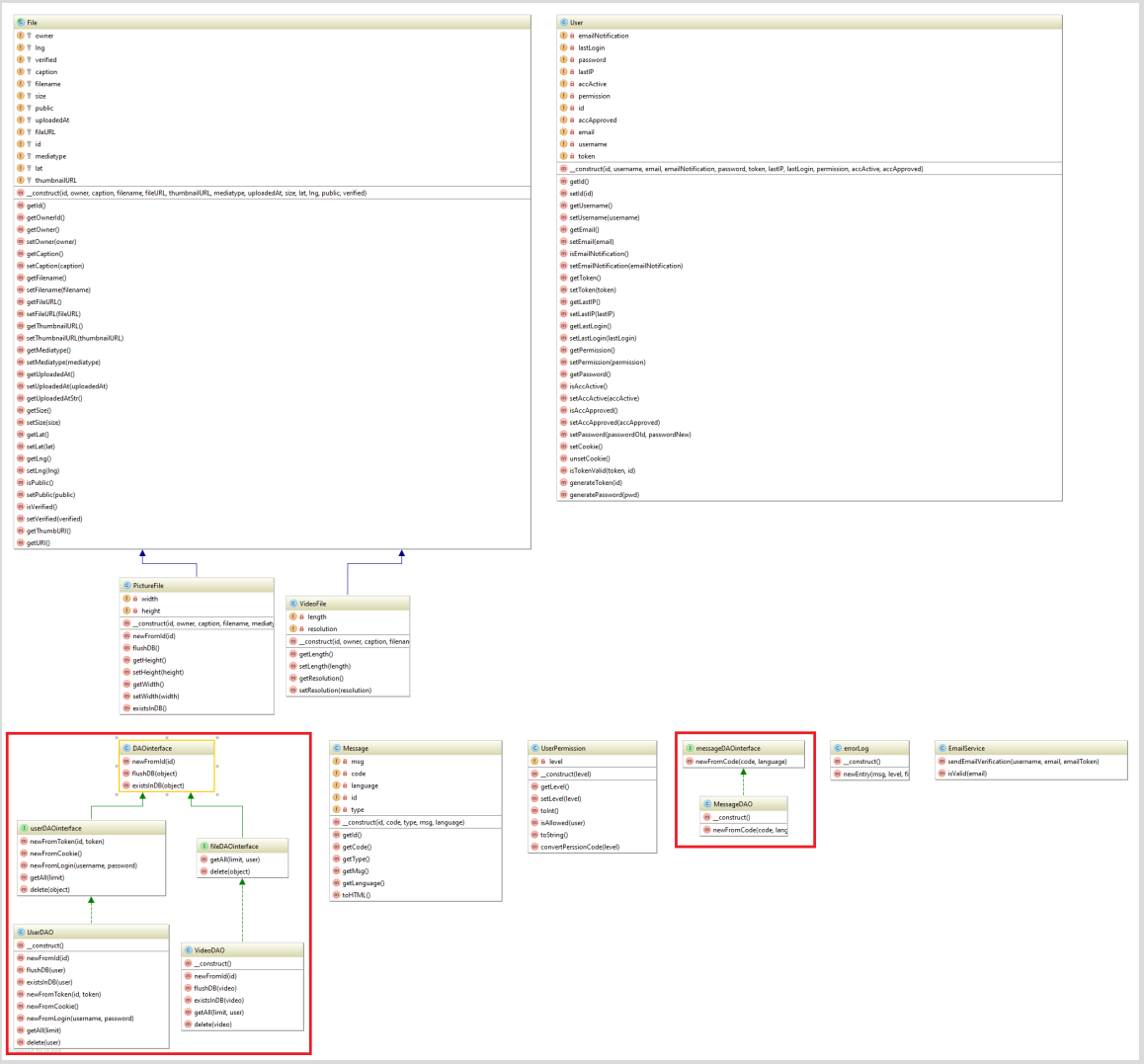
this weeks task was to refactor a part of our project code to implement a Design Pattern. We have chosen to implement the Data Access Object Pattern (DAO Pattern) in our server-side code. We have used it to encapsulate our low-level database communication from our server logic. The DAO Pattern is made up of

- *Model Objects* which represent one database entry,
- A *Data Access Object Interface (DAO Interface)* which defines standard actions to be performed on the *Model Objects* and
- the *Data Access Object* itself which implements the *DAO Interface* and is responsible for the communication with the data storage.

The following screenshot shows our **old** implementation of our Backend logic:



After refactoring our code the class diagram looked like this (the **new** implementation):



You can click on both pictures to open them in full resolution.

Have a nice Sunday,

team Unveiled

28. April 2016 by CodeLionX on Allgemein

HW15: Fowler

refactoring

Hey there,

this weeks homework was to work through chapter 1 of Fowler's book *Refactoring* and doing the steps he did in the example. We would like to share our Github-Repos in which we documented our refactoring steps. Use the following links:

- CodeLionX: Fowler/refactor
- SeanAda: Fowler/master
- Digister: Fowler-Software-Engineering-Homework/master

We hope you like our work and wish you a nice rest of the week.

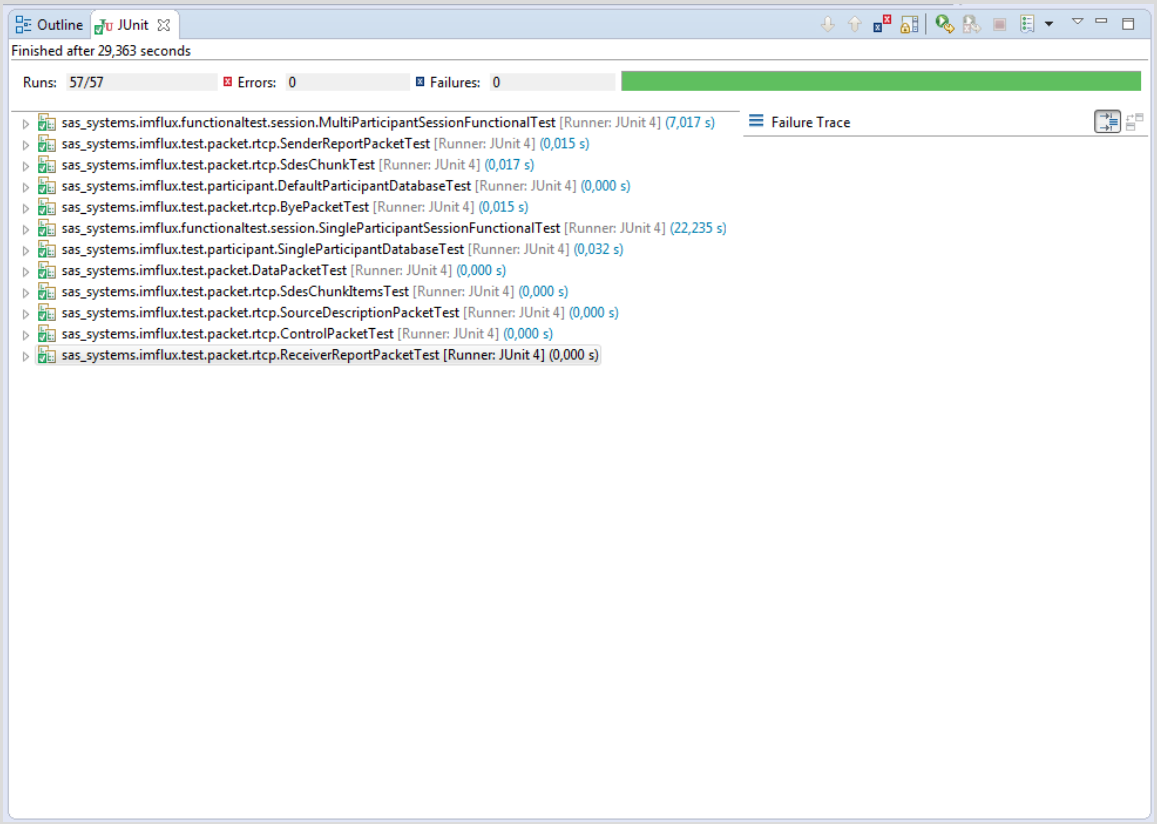
Greetings

– team Unveiled

25. April 2016 by CodeLionX on Allgemein

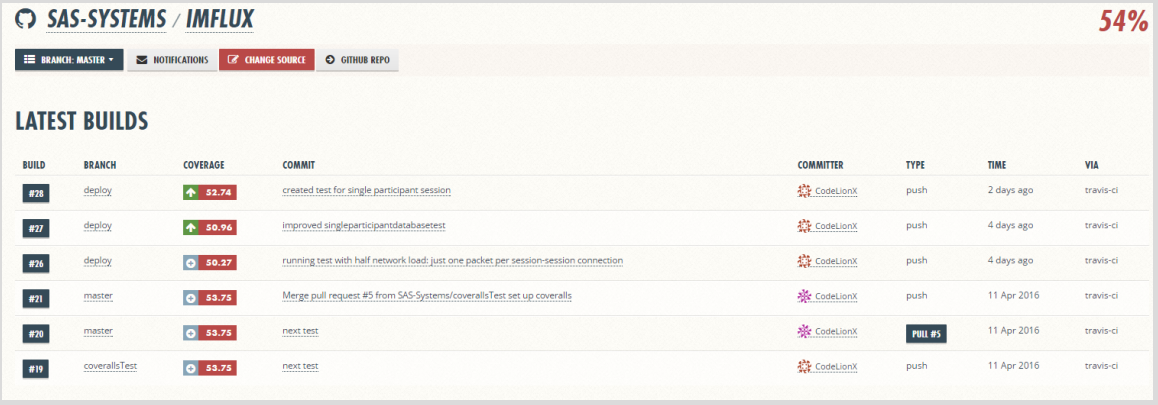
HW14: Unittesting

With this blog entry we want to share our testing approach. We are using JUnit for testing our backend. You can find our Testing code on Github and the following screenshot shows our IDE running these tests:



We are also using Travis-CI and Coveralls for running our tests automatically. Our build-tool is Maven and we use it to perform our testing on Travis. You can find our Maven build-file here. See the below screenshots:

```
1528 [INFO] Starting Coveralls job for travis-ci (124033212)
1529 [INFO] Git commit 2ea024e in deploy
1530 [INFO] Writing Coveralls data to /home/travis/build/SAS-Systems/imflux/target/coveralls.json...
1531 [INFO] Processing coverage report from /home/travis/build/SAS-Systems/imflux/target/site/jacoco/jacoco.xml
1532 [INFO] Successfully wrote Coveralls data in 525ms
1533 [INFO] Gathered code coverage metrics for 42 source files with 8269 lines of code:
1534 [INFO] - 2264 relevant lines
1535 [INFO] - 1194 covered lines
1536 [INFO] - 1070 missed lines
1537 [INFO] Submitting Coveralls data to API
1538 [INFO] Successfully submitted Coveralls data in 848ms for Job #28.1
1539 [INFO] https://coveralls.io/jobs/13689600
1540 [INFO] *** It might take hours for Coveralls to update the actual coverage numbers for a job
1541 [INFO]     If you see question marks in the report, please be patient
1542 [INFO] ------------------------------------------------------------------------
1543 [INFO] BUILD SUCCESS
1544 [INFO] ------------------------------------------------------------------------
1545 [INFO] Total time: 46.180 s
1546 [INFO] Finished at: 2016-04-18T21:48:50+00:00
1547 [INFO] Final Memory: 27M/491M
1548 [INFO] ------------------------------------------------------------------------
1549
1550
1551 Done. Your build exited with 0.
```



Demo Application in TDD

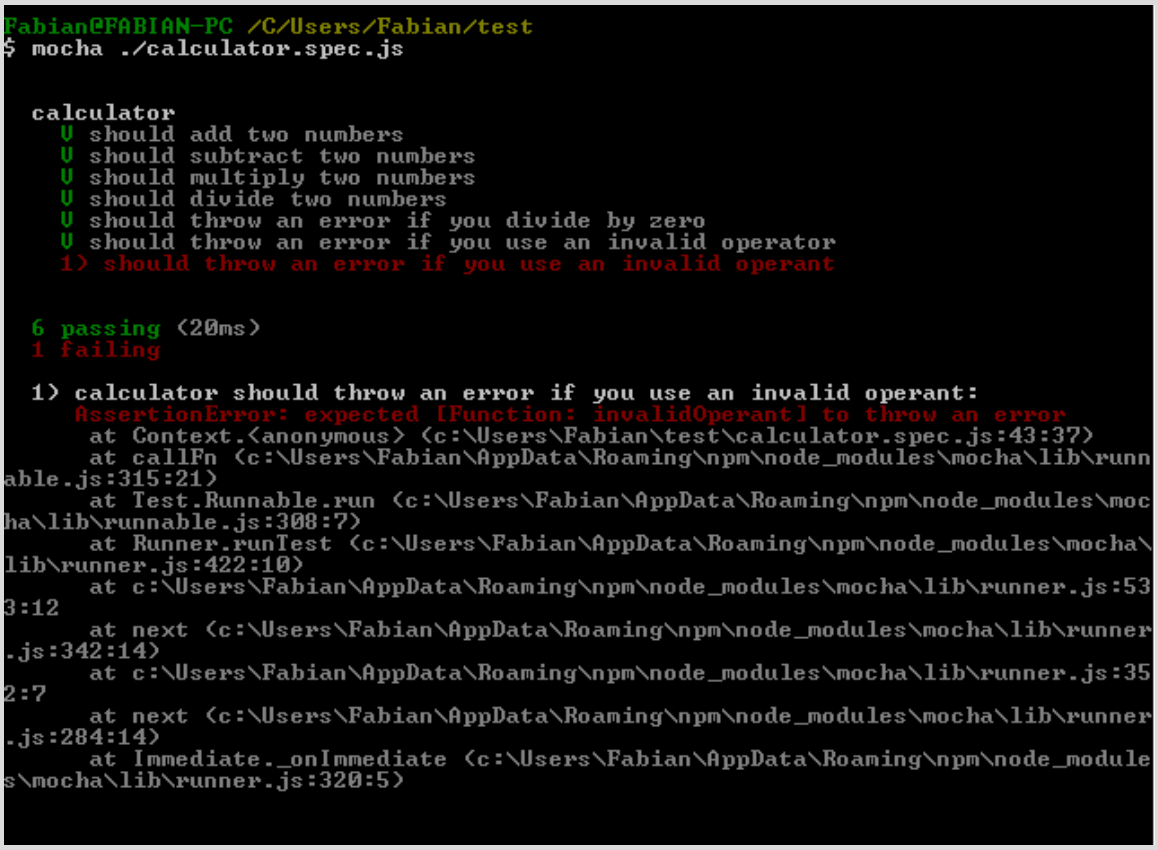
First of all we used Node.js to install the Mocha and Chai javascript Framework for testing. Now that we have access to the frameworks we were able to write our first tests and implement the function afterwards. For Node.js there is no need of an IDE, therefore we just used Sublime Text 2 and the command console.

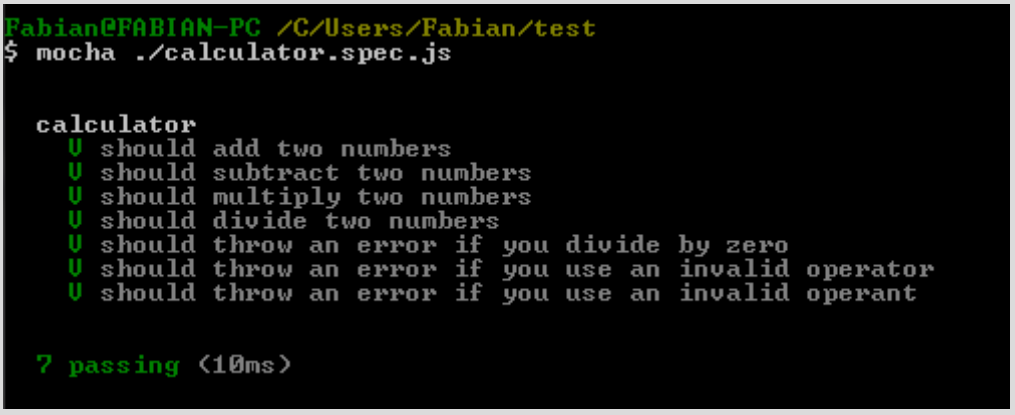
An advantage of Mocha and Chai is that the test is really easy to read. For example:

```
describe('calculator', function(){
    it('should add two numbers', function(){
        var result = calculate('3', '+', '5')
        expect(result).to.equal(8)
    })
    ...
});
```

In this test we describe the calculator. A special case of this test is that it(the calculator) should add two numbers. Now we call the calculate function with the operants 3 and 5 and the operator +. Because 3 + 5 = 8 we also expect that the result equals 8.
If you run mocha now the test will fail, because there is no code which will be executed. Now we implemented the `calculate()` – function to fix this problem. When we are done with fixing this special problem, we are going to repeat this pattern. This means we write a new test and fix it with enhancements of the `calculate()` – function.

You can find the final code in our Documentation-repository and the results of the tests here:





o

15. April 2016 by CodeLionX on Allgemein

Backend API Documentation

Postman API

We have tested our API with the Google Chrome Plugin Postman. You can add our Collection of Tests to your Postman to see and run the tests as well. Use this link.

14. April 2016 by CodeLionX on Allgemein

HW13: FP calculation and time estimation (depricated)

Hi folks!

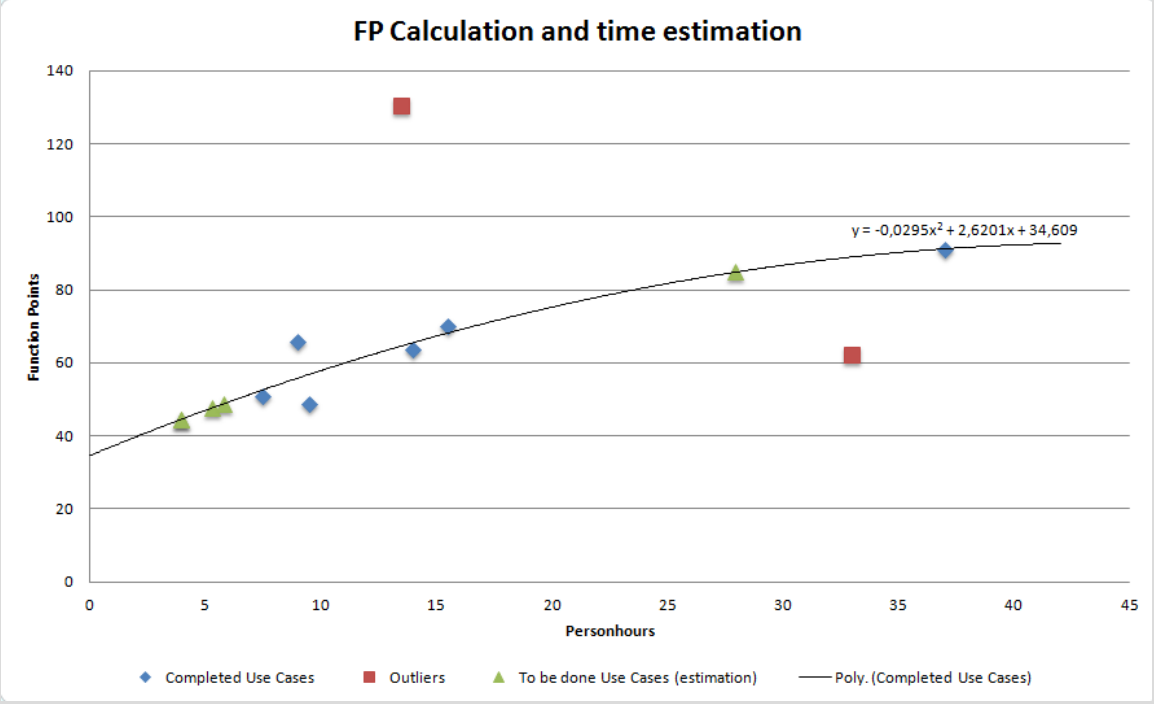
The following table shows our already completed use cases with our time spent implementing it and the corresponding Function Points:

Use Case	Use Case Name	Total Time Spent (Estimation in h)	Function Points
(1	Capture and Stream Video	13,5	130,38)
2	Configure Settings	15,5	69,96
3	Maintain User Profile	9	65,72
4	Switch User	7,5	50,88
5	Register	14	63,60
6	Browse Media	33	62,40
7	Manage Users	9,5	48,76

General	RTSP Library	42	91,16

We plotted this data in a graph with the Function Points on the ordinate and the total time spent on the abscissa. As you can see we have two outliers plotted in red. The first one at x=13,5 is the Use Case Capture and Stream Video. It is very complex and affects various files, therefore it has very much FPs. We have not completed this Use Case yet and have estimated a lower number of hours, because most of the time will go to the streaming library. Maybe this was a mistake and we have to correct our estimation, so that it matchs with our chart.

The second outlier is the Use Case Browse Media, where we have spent too much time in spite of the low number of FPs. The reason for that was a change in our architecture and the technology we used for creating a web-based media browser. This lead thereto that we had to implement most of the functionality again.



We used this graph to estimate our remaining Use Cases. You can see these estimations as green triangles in the graph.

You can find all our Use Case Specifications here: Documentation

Have a nice evening,

your team Unveiled

EDIT:

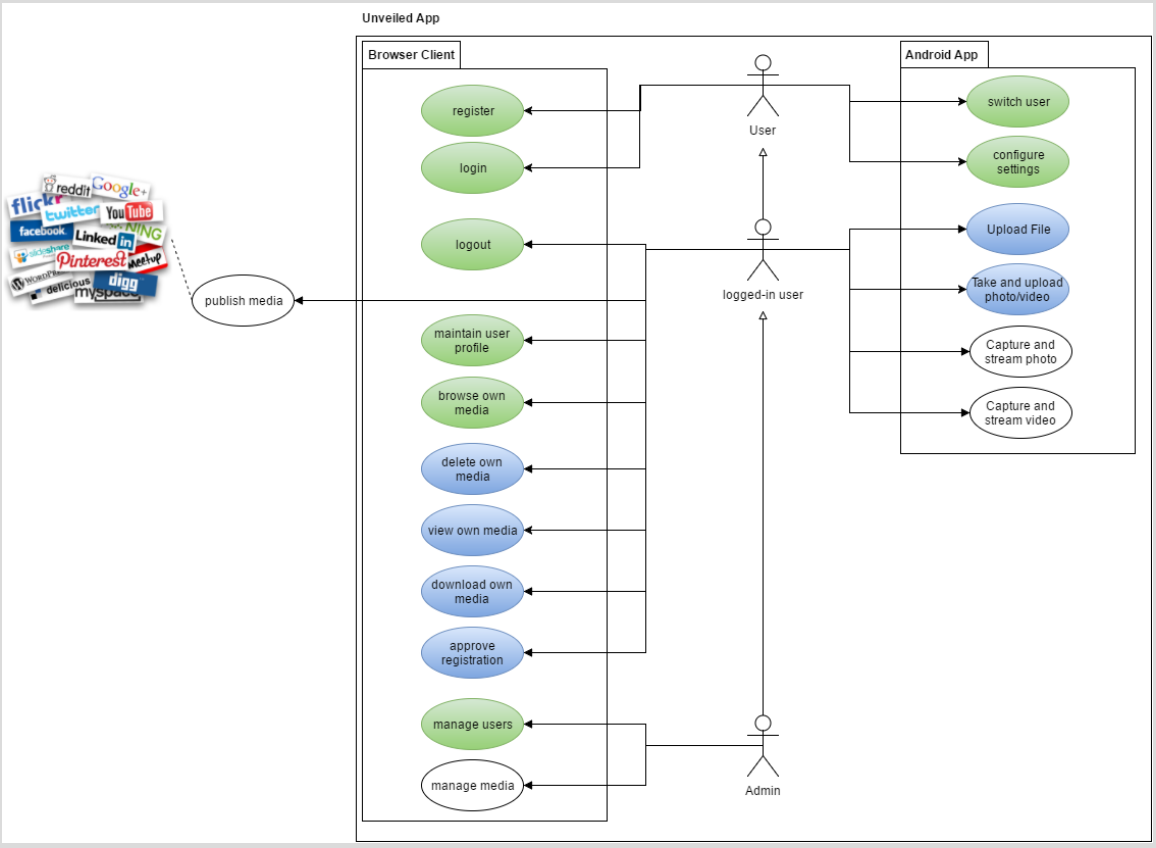
There is a new version of the function point calculation and Use Case estimation. Please go to this document.

4. April 2016 by CodeLionX on Allgemein

HW12: Scope for 2nd Semester and Risk Management

Hey there,

this is our new overall Use Case Diagram:



The green marked Use Cases were finished in the last semester and the blue ones show the scope for this semesters work on the Unveiled project. You can find all our Use Case specifications on our Documentation site. These are the new ones (the blue marked Use Cases):

- Use Case: Delete own Media
- Use Case: Download own Media
- Use Case: View own Media
- Use Case: Approve Registrations
- Use Case: Upload File

Risk Management

We used our Google Drive folder to create a new dynamic spreadsheet which contains our list of possible risks. You

can see the current status of the list below or in the document:

Risk Rank	Risk Name	Risk Descri-ption	Risk Proba-bility of Occurence	Risk Impact	Risk Factor	Risk Mitigation	Person in Charge of Tracking
1	RTSP library won´t finish	We are writing our own streaming library based on an existing approach. This is more complex than expected and may take longer to implement	60%	100%	60%	Don't use streaming to transmit the data -> use standard upload/download	Fabian Schäfer
2	lib streaming doesn´t work	The Android App will make use of an external librar	50%	90%	45%	check dependencies and architecture early; use own librar	Sebastian Adams

			y called “libstreaming”; when it does not fit into our architecutre it will not work				y for streaming		
	3	exam preparation disrupts project progress	in final project phase also exam preparation takes place, therefore team members must prioritize and can not spend all their time for the project	100%	40 %	40 %	Make a detailled planning for this final project phase early enough; cut off requirements	Sebastian Schmidl	
	4	server contract expires	Server contract expires or 1&1	80%	40 %	32 %	check contract periodically; search	Sebastian Adams	

			termi nates the contra ct				and prepa re backu p solutio n	
	5	bad code qualit y	end produ ct contai ns a lot of bugs or the perfor mance of the applic ation is poor	40%	60 %	24 %	Write tests; make code revie ws	Fabian Schäfer
	6	case of illness	A team memb er sustai ns a little or seriou s injury or becom es sick for a longer time period	40%	30 %	12 %	divide tasks of this perso n to the other team memb ers; cut off some requir ement s	Sebasti an Schmid l
	7	server stops worki ng	Privat e hosted server stops worki ng, becau se of	30%	30 %	9%	invest additi onal time on gettin g server runni	Sebasti an Adams

		any issue				ng again; make data backups periodically; provide user guide with server set up steps	
8	team member leaves the university	Team member becomes exmatriculated or leaves the univerity beacause of personal reasons	10%	70 %	7%	cut off requirements and use cases	Sebastian Schmidl

Use Case Estimation

The following table shows our time spent for the different Use Cases without all the work which would belong to multiple Use Cases:

Seme	Use	Use Case	Time spent (estimation in	Func

ster	Case	Name	h)			tion Points
			Documentation	Coding	Testing	Total	
2	1	Capture and Stream Video	1,5	8	4	13,5	
1	2	Configure Settings	1,5	8	6	15,5	
1	3	Maintain User Profile	1	6	1	8	
1	4	Switch User	1	4	2,5	7,5	
1	5	Register	1	8	1	10	
1	6	Browse Media	1	34	2	37	
1	7	Manage Users	1,5	6	2	9,5	

Page 1 of 3 Older Posts →

Documentation links:

- SRS
- SAS
- Use-Case: Capture and stream video
- Use-Case: Configure settings
- Use-Case: Maintain user profile