Azer Hojlas
CINTE-19
Modern mjukvaruutveckling, IV1303

Assignment 1

Overview:

As I am working alone, I will be both the client and team leader.

Client requirements:

- Implementation of a calculator that supports addition, subtraction, multiplication and division.
- Client-product interaction will be through the command prompt.

Opportunity:
Instead of tedious handwritten calculations, especially for the more labour-intensive ones', it is much more cost effective to use a program that performs these duties for the clients. This outsourcing of basic calculations will be a great return on investment for the client. The team-leader assesses that the requirements can be implemented given the scope of the course IV1303.

Requirements:

Format 1

Requirement 1

Description of the requirement	**Rationale**	**Reference**	**Source**
A calculator that supports the 4 basic mathematical operations.	Calculations that are faster to compute and with less inherent risk of errors due to the human factor	Not necessary	For the client, by the producer

Requirement 2

Description of the requirement	**Rationale**	**Reference**	**Source**
Client-product interaction through the command prompt.	Easy to use and without the need to adapt to a new GUI, instead the client uses the console which most people are familliar. with.	not necessary	For the client, by the producer

Format 2

Requirement 1

User story 1.1

As a user I want to use the calculator in order to get the result of the four basic mathematical operations with two integers.

Requirement 2

User story 2.1

As a user I want to access the calculator through the console in order for it to be easily accessible and easy to adapt to, instead of having to adapt to a new GUI.

Format 3

Priorities format: As the alphabet progresses, the priority of the requirement decreases, i.e A has the highest priority and Z the lowest.

Requirement 1

Use Case ID	**Pre-condition**
01	The PC has JRE installed The PC has a pre-compiled version of the program, i.e a class-file. The client knows how to use the four basic mathematical operations beforehand
Use Case Title	**Post-condition**
Calculator	The result of a basic mathematical operation done on two integers, which can be a floating point number depending on the operation.
Actors	**Assumptions**
The client, their PC and everything with the accompanying resources for a PC, e.g the console.	The PC has a operating system that supports JRE The software will support the four mathematical operations. Operator “+” refers to addition. Operator “-” refers to subtraction. Operator “/” refers to division. Operator “*” refers to multiplication.
Flow of Events	**Priority**
● The program is started by the client in a console. ● the client writes in specific commands in the calculator. ● The commands are written in the format: java calculator <integer> <mathematical operation> <integer> . ● The client presses “ENTER” to execute the calculation. ● The calculator confirms that the inputs are correct and follow a correct format. ● The calculator performs the operation and	9

outputs the result to the terminal, either an integer or a floating-point number.	
Alternative Events	
There are no alternative ways to run the program. If a error occurs like non-integer inputs or not recognized operators, a exception will be thrown and the client will be notified in the console.	

Requirement 2

Use Case ID	**Pre-condition**
02	The PC has JRE installed The PC has a pre-compiled version of the program, i.e a class-file. The client knows how to use the four basic mathematical operations beforehand A running console
Use Case Title	**Post-condition**
Command prompt interaction	A correct output
Actors	**Assumptions**
The client, their PC and everything with the accompanying resources for a PC, e.g the console.	The client knows how to use the command prompt.
Flow of Events	**Priority**
● The client navigates to the class-file thorugh the terminal ● The client executes command: java calculator <integer> <operator> <integer> ● The client will receive the correct result	B
Alternative Events	
If the arguments are incorrect then the program will notify the client of what exactly they did wrong	

Format 4

Priorities format: As the alphabet progresses, the priority of the requirement decreases, i.e A has the highest priority and Z the lowest.

Requirement 1

General Requirement Description
Requirement ID: 01
Requirement Title: Calculator
Requirement Description: A calculator that supports the four basic mathematical operations.
Requirement Type: Main functionality of the program.
Internal/External Requirement: External requirement.
Rationale: Outsourcing of calculations to a program is time- and energy efficient.
Event/Use Case ID: 01
Related To Requirement(s): Requirement 2.
Conflicting Requirements: None
Non-Functional Requirement(s): None
Constraints: The software can only be run on a computer
Intended User: The client or user doing the calculations
Customer Satisfaction: A fundamental requirement that is required in order for the customer to be satisfied.
Customer Dissatisfaction: The customer will be completely dissatisfied if this requirement is not met.
Assumptions: The computer must have JRE installed and the client must follow the command format which is: java calculator <integer> <operator> <integer>
Reference Document(s): None

Requirements Evaluation Data
Business Value: High.
Other Value: None
Requirement Priority: A
Fit Criterion/Criteria: Capable of performing the basic mathematical operations.
Risk(s): None

Other Description Data
System Data: Must be run on computers with the same OS as they were compiled on.
Interfacing System ID: None
Environment: Computers that support JRE

Requirement 2

General Requirement Description
Requirement ID: 02
Requirement Title: Command prompt interaction
Requirement Description: The client interacts with the program through the command prompt.
Requirement Type: Main functionality
Internal/External Requirement: External

Rationale: States how the client will use the software.
Event/Use Case ID: 02
Related To Requirement(s): Requirement 1
Conflicting Requirements: None
Non-Functional Requirement(s): None
Constraints: The client must follow the command format which is: java calculator <integer> <operator> <integer>, otherwise the software will not function properly
Intended User: Users that would otherwise perform calculations by hand
Customer Satisfaction: Requirement is imperative for the software to function
Customer Dissatisfaction: The client has specifically asked for this exact implementation
Assumptions: The client must know how to use the console.
Reference Document(s): None

Requirements Evaluation Data
Business Value: High.
Other Value: None
Requirement Priority: B
Fit Criterion/Criteria: The clien tmust know how to run the software on the command prompt.
Risk(s): None

Other Description Data
System Data: None, works the same on all systems
Interfacing System ID: None
Environment: Computers that support JRE and has a console

Alpha cards:

Opportunity

Identified check-list:

Idea behind opportunity identified: OK, The client and the producer recognize that the development of a calculator would save the client time and energy, and ultimately, money.
At least one investing stakeholder interested: OK, The client has ordered the software
Other stakeholders identified: OK, the developer

Requirement Item

Identified:

1. *Requirement Item is briefly described, OK.*
2. *The Requirement Item is logged, OK.*
3. *The origin of the Requirement Item is clear, OK*

Requirements

Conceived:

Stakeholders agree system is to be produced: OK, the client has made an order for the system.
Users identified: OK, the client is the user.
Funding stakeholders identified: NOT OK, unclear whether the client is financing the project with debt or equity.
Opportunity clear: OK, Opportunity has been identified in the opportunity alpha.

Check-list item that was difficult to determine:
It was difficult to identify the funding stakeholders, as the developer does not know if the client is using equity or debt to finance the project. In the first case they are paying for it by themselves, and in the other debt has been financed either through borrowing from financial institutions or the issuance of bonds as is common for private companies.

Result:

Assignment completed successfully.

Assignment 2

Test Case ID	List of modules
01	Requirement 1 Requirement 2
Input	Output
3 + 2 3 / 2 -3 * 2 3 - 2	5 1.5 -6 1
Environmental needs	Inter-case depencies
PC has to have JRE installed and a functioning console	None
Special procedural need	
<java calculator> has to be written in to the command prompt before the input	

Test Case ID	List of modules
02	Requirement 1 Requirement 2
Input	Output
s - 2 2 - s 3 : 2 3 + 2.0 1 / 0	“Error, argument 1 is not an integer” “Error, argument 3 is not an integer” “Error, argument 2 is not a valid operator” “Error, argument 3 is not an integer” “Error, division with 0 is not allowed”
Environmental needs	Inter-case depencies
PC has to have JRE installed and a functioning console	01 has to function in order for 02 to be valid

Special procedural need	
<java calculator> has to be written in to the command prompt before the input	

Planning poker estimation

- As I work alone I have nobody to play planning poker against, but I have estimated that it should take about 30 minutes to implement and test the two requirements and tests.

Regular estimation

Division of requirement 1 and 2 into low-level requirements
(The second requirement is so small that it's implementation cannot take more than a few minutes):

- The calculator must be able to compute two integers with the four basic mathematical operations.
- The integers and the operator will be inputted through the terminal.
- The number arguments must be of the type integers
- The operator argument must be one of the following: '+', '-', '/' or '*'.
- Addition, multiplication and subtraction output a integer
- Division may output a floating point number

LOC-estimation:

We have three arguments. Assume that each of the following actions require two lines of code per input:

- Each argument needs to be checked if it is a valid input (LOC += 3*2)
- Each argument has to be assigned to a variable (LOC += 3*2).
- The operator has to be checked four times for which operation that needs to be performed (LOC += 4*2).
- Operation is performed (LOC += 1*2)
- An output value is assigned (LOC += 1*2)
- The output value is printed out (LOC += 1*2)

According to the estimations above, the total LOC would be 26. I estimate that this would take about 20 minutes to implement, including debugging.

Comparing traditional and agile ways of estimation
I personally prefer the agile way of estimating time and size. The traditional way however, has more stringent rules for how to estimate man-hours and LOC. This may result in a smaller margin of error but is also more time-consuming, which is the main reason for my preference of the agile way.

Status of software system alpha

Architecture Selected
- OK

Demonstrable
- OK

Usable
- OK

Operational
- OK

Demonstrable
- OK

Ready
- OK

Retired
- NOT OK, system has not been discontinued.

Status of team alpha

Seeded:
- OK

Formed:
- OK

Collaborating:

- OK

Performing:

- OK

Adjourned:

- NOT OK, mission not yet concluded

Status of requirements Item alpha

Identified

Requirement 1:

- OK

Requirement 2:

- OK

Described

Requirement 1:

- OK

Requirement 2:

- OK

Implemented

Requirement 1:

- OK

Requirement 2:

- OK

Verified

Requirement 1:

- OK

Requirement 2:

- OK

Status of requirements alpha

Conceived:

- OK

Bounded:

- OK

Coherent:
- OK

Acceptable:
- OK

Addressed:
- OK

Fulfilled:
- OK

Status of opportunity alpha

Identified:
- OK

Solution Needed:
- OK

Value Established:
- OK

Viable:
- OK

Addressed:
- OK

Benefit accrued:
- OK

Result:
System has been created and all alphas thus far have been fulfilled.

Assignment 3

Opportunity alpha
I have understood the conclusions of the various team members, I do not however agree with that they are on the value established alpha. They all seem to be in agreement that a solution has been identified. However, they are unsure of it, as Cynthia points out: "The desired outcomes are clear. But, we don't yet know all the risks and we don't yet know if we can develop and deploy this new system within the constraints we have, including the schedule and budget constraints that have been given to us".

Stakeholders alpha
I have nothing to add on this alpha.

Requirements Alpha
This group discussion was somewhat lengthy for me. I had a difficult time of following their thought process. This was maybe something I as a reader should have understood better beforehand, nevertheless I would have liked to know how exactly they achieved the other points on the list of the bounded state.

Software System Alpha
I agree with Cynthia. I do not however understand the reasoning of Sam and Fred as to why they both raised the Architecture Selected card. They must have some reasoning behind this which is not stated, or at least some counter-arguments to Cynthia.

Team Alpha
I can only agree with their discussion.

Work Alpha
Cynthia never stated her opinion.

Way of Working Alpha
Maybe other stakeholders should have been present during this meeting or assigned beforehand.

Assignment 4

Risk identification and analysis
The previous estimations of assignment 2 may prove to be wrong, however not by a large margin because this software takes very little effort (size) and time to implement.

Team programming

As I have been working alone, I have made a literature study to find out the good and the bad sides of pair-programming using at least two references.

> I have listed the good sides expressed at the website of a organization called "agile allegiance":
>
> - increased code quality: "programming out loud" leads to clearer articulation of the complexities and hidden details in coding tasks, reducing the risk of error or going down blind alleys.
> - better diffusion of knowledge among the team, in particular when a developer unfamiliar with a component is pairing with one who knows it much better.
> - large reduction in coordination efforts, since there are N/2 pairs to coordinate instead of N individual developers
>
> I have listed the bad sides expressed by known IT-journalist named Ritesh Kumar:

- Decrease in productivity : It happens with some people that they can’t work or concentrate while someone is watching them or interrupting them. Working in a team of two generally becomes a social affair and you end up wasting time on talks , chit chats and fun.
- Friction and ego : There can be scenarios where yours thought will conflict your teammate’s. The egos of two developers might collide which can put you to standstill and ruin the relationship amongst team mates.

Test-first programming

Pros:

- Comfort of knowing what has to be achieved beforehand
- Can be executed as soon as the implementation of software is done

Cons:

- Might be difficult to define test-cases beforehand
- The requirements might change as the work moves along, rendering the test cases useless.

Refactoring

Pros:

- Improves the design of the software and makes it easier to understand
- Improves efforts to debug the code

Cons:

- Time consuming.
- The cost of refactoring may be higher than rewriting the code from scratch.

Designing the code first before coding

Pros:

- I got a better feel for how large the project was going to be and how much time I would need.
- Instead of random coding, I structurally did what I had planned beforehand.

Cons:

- Was a tad bit time consuming for such a small project. It is however most certainly a time saver for really long projects that might take months to complete.

Estimations VS actual implementation

My estimations were a bit off, both in LOC and time spent coding. I had predicted that my code would only have 26 LOC, while in reality it was 67, making my estimates not so reliable. It also took roughly double the time to implement than what was stated in assignment 2. All in all I was a bit too optimistic about my prospects.

Actual code:

```
public class calculator {

   public static void main (String[] args) {

      int integer1;
      String operator;
      int integer2;
      double output;

      try {
       Integer.parseInt(args[0]);
      } catch (NumberFormatException num1) {
          System.out.println("Error, argument 1 is not an integer");
          System.exit(1);
      }

      try {
       Integer.parseInt(args[2]);
      } catch (NumberFormatException num1) {
          System.out.println("Error, argument 3 is not an integer");
          System.exit(1);
      }

      try {
          if ((!args[1].equals("/")) && (!args[1].equals("*")) &&
(!args[1].equals("+")) && (!args[1].equals("-")))
           throw new Exception();
      } catch (Exception op) {
          System.out.println("Error, argument 2 is not a valid operator");
          System.exit(1);
      }

      integer1 = Integer.parseInt(args[0]);
      operator = args[1];
      integer2 = Integer.parseInt(args[2]);

      try {
          if (args[1].equals("/") && (integer2 == 0))
           throw new ArithmeticException();
```

```
        } catch (ArithmeticException ar) {
            System.out.println("Error, division with 0 is not allowed");
            System.exit(1);
        }

        if (args[1].equals("/")) {
             output = (double) integer1 / integer2;
             System.out.println(output);
        }

        if (args[1].equals("*")) {
             output = (double) integer1 * integer2;
             System.out.println((int) output);
        }

        if (args[1].equals("-")) {
             output = (double) integer1 - integer2;
             System.out.println((int) output);
        }

        if (args[1].equals("+")) {
             output = (double) integer1 + integer2;
             System.out.println((int) output);
        }

    }
}
```

Assignment 5

Alpha Cards:

Requirement alpha:

- Conceived state: OK
  The stakeholders are the students, administrators and the faculty members.

- Bounded state: OK
  The requirements have been bounded. Requirements, assumptions etc have been set, e.g the capacity for the system should be 5000 users.

- Coherent state: OK
  The conflicts that existed have been addressed. The group has parleyed with the

dissatisfied faculty member and taken their issues into account.

- Acceptable state: NOT OK
  The system is not finished and additional features are required, which we can deduce from the complaints of the faculty members in the following paragraph:
  "Through the interviews and demonstration, the team realized that the missing features requested by the faculty members were related to the way the new solution computes grades and manages feedback on deliverables. The new system is more restrictive. It does not allow faculty members to associate grading components to each student deliverable and it only supports grades based on points, not on alphabetic symbols."

Team alpha:

- Seeded state: OK
  Mission has been defined, as have the other items on the list, like for instance the constraints, size, governance, leadership etc.

- Formed state: OK
  All items on the list seem fine to me.

- Collaborating: NOT OK
  My assessment is that the team members are not sufficiently acquainted to each other. Minor differences have led to a lack of open and honest communication, which can be best illustrated in the following paragraph: "After the interviews, two developers were in disagreement about one significant requirement. They shared the conflicting viewpoints with the faculty involved. The faculty agreed with the first developer, and the second developer felt somewhat put out that his opinion did not seem to matter. Since this was not the first time that the second developer's ideas had not been accepted, little by little he stopped communicating with the team." As we can see, one developer does not feel as appreciated and therefore sees no reason in communicating as their opinions seem to not be worth much.