Sprint reflection 4

Context Project: Health Informatics
Group: HI4

User Story	Task	Task Assigned To	Estimated Effort per Task	Actual effort per task	Done (yes/no)	Notes
The user wants to see the results of the files that are read or analyzed.	- Extend the GUI so that the output is shown - Format the output in the right way - Update the writer so it can use a TreeSet	Remi Elvan Elvan	3 3 3	3 3 3	Yes Yes Yes	Output is visible and can be saved in the specified format
The user wants to add labels to created records, so analysing is easier by mentioning labels instead of records.	- Create a mechanism to add labels to only the data records that comply with the condition. - (Create conditions language element)	Matthijs Matthijs	6 This was not included in the sprint plan	7 5	Yes Yes	Invested five extra hours in creating an extendable conditions parser.
The user wants to perform computations (such	- Extend language construct for computing	Elvan	2	2	Yes	Examples can be found in the user manual

as counting, summation or statistical operations) on the data for exploratory analysis.	- Implement (script) code for computations on the data) - (Create conditions language element)	Elvan Elvan	5 This was not included in the sprint plan	5 5	Yes Yes	A few could haves are implemented: deviation, variance, sum of squares
The user wants to know how to use the scripting language	- Create examples and descriptions in a manual	Sven	5	5	Yes	Also did some acceptance testing with Wenxin
The user wants to perform data transformations by entering a scripting language in a script editor	- Improve the script editor by implementing the RichTextFX library - Improve the script editor visually - Display which data is available in a tree list view	Remi Remi Sven	6 2 4	6 2 4	Yes Yes Yes	This is still not fully linked with the GUI at the moment (could not be shown at the demo)
The user wants the scripting language to be understood by the program in order to	- Implement basic structure for recognising the C's - Explain the basic structure for recognising the C's - Parse chunking - Parse constraints	Hans Hans Hans Sven	6 2 6 6	6 2 4 4	Yes Yes Yes Yes	A generic parser and subparsers for chunking, constraints, coding and computations

	- Detecting syntax errors before running the script	Hans	4	4	Yes	were implemented
The user wants to import files with the least amount of effort	- Automatically detect column names if they are in the file - Let the user rearrange columns (and their names) by clicking and dragging	Matthijs Matthijs	4 4	4 4	Yes Yes	The user can indicate if the first line of the input file contains the column names
The client wants an up to date emergent architecture document	- Update the architecture document	Elvan	2	2	Yes	Added some explanation on FXML and MVC
The user wants a reactive GUI with error management (including dialog windows and error messages)	- Test the GUI on JavaFX Services with JUnit tests following the dependency injection pattern	Remi	4	-	No	This will be moved to the next sprint (as part of the refactoring task for SIG input)
The user wants the program to be fully tested and an explanation on warnings that are ignored	- Report why some warnings from Cobertura, PMD, FindBugs and CheckStyle don't have to be solved	Sven	2	2	Yes	This file is included in the SE Deliverables folder on the repo

Main Problems Encountered

Problem 1

Description: Underestimated implementation time for the parser

Reaction: We decided to write our own parser instead of using a library. We knew that this would take more time than using a parser generator but we think that the advantages of a hand-written parser outweigh the extra implementation time. An important advantage of writing your own parser is that you can easily write specifications and provide high-quality error messages on syntax errors (with error correction). This is also why we planned extra hours for the implementation of the (generic) parser but the implementation of the subparsers, which are specified on the eight C's, took more time than expected. This was at the expense of testing the GUI. Luckily, now the parser is implemented in such a way that it can easily be extended for other data transformations in the future. Therefore, we can fully focus ourselves on GUI testing and new functionality in the next sprint.

Adjustments for the next Sprint Plan

(Motivate any adjustments that will be made for the next Sprint Plan.)

This week went really well. Besides the problems with the GUI tests, we finished all the tasks from the sprint plan and even implemented a few extra features, such as statistical computations (average deviation, sum of squares, variance) and subparsers for the computation and coding operations. We also did some extra testing on the reader and writer.

- Add a refactoring item to the sprint

Because the initial input for SIG has to be delivered next week Friday, we will add a refactoring task to the next sprint. This means that everyone has to make sure the code he or she has written up till now is fully tested (coverage should be at least 75%) and is checked by Cobertura, PMD, FindBugs and CheckStyle. Also a testing task for the GUI will be added to the sprint just like this week.