Martijn Wils

Requirements list: classification tree tool

[Applicatie 1](#_Toc366665026)

[1 Business requirements 1](#_Toc366665027)

[2 Functional requirements 2](#_Toc366665028)

[3 Design functionaliteiten 3](#_Toc366665029)

[4 Performance 3](#_Toc366665030)

[5 Usability 4](#_Toc366665031)

[6 Optionele functionaliteiten 5](#_Toc366665032)

[7 Scope (sprints scrum) 6](#_Toc366665033)

[7.1 Sprint 1 6](#_Toc366665034)

[7.2 Sprint 2 7](#_Toc366665035)

[7.3 Sprint 3 8](#_Toc366665036)

[7.4 Sprint 4 9](#_Toc366665037)

[7.5 Sprint 5 en 6 10](#_Toc366665038)

[7.6 Overige functionaliteiten 11](#_Toc366665039)

# Applicatie

Het doel van deze applicatie is om een duidelijke en gratis tool te hebben waarmee classification tree testing mogelijk is. Deze tool wordt hoofdzakelijk gebruikt voor opleidingen en kan mits voldaan aan de eisen overgedragen worden aan de leerlingen na hun opleiding.

# Business requirements

1. De applicatie moet simpel te gebruiken zijn, nieuwe gebruikers moeten snel hun weg kunnen vinden binnen de applicatie.
2. De applicatie moet een positief beeld weergeven over Improve (logo geïmplementeerd). De applicatie zal namelijk gebruikt worden door de medewerkers van Improve, maar ook door leerlingen die deelnemen aan een opleiding die gegeven werd door een medewerker binnen Improve.
3. De applicatie moet gemakkelijk te onderhouden zijn na de periode van 29 november (einde stage). Dit door middel van versiebeheer (github) en duidelijke code conventions.
4. De applicatie zal ontwikkeld worden in 6 sprints over een tijd van 12 weken.
5. Na 12 weken moet de applicatie stabiel zijn om te gebruiken voor eigen doelstellingen of voor presentaties.
6. De applicatie moet in het Engels zijn.

# Functional requirements

1. Het moet mogelijk zijn om de inhoud van de file te saven en terug te openen
2. De applicatie moet in staat zijn meerdere files te openen: Multi-document interface.
3. Een file moet een classification tree bevatten. Met andere woorden: mag niet leeg zijn.
4. Elke nieuwe file begint met het automatisch aanmaken van een root.
5. De gebruiker van de applicatie moet de mogelijkheid hebben om onderverdelingen te maken op de classification tree.

1. Bij het aanmaken van een classification tree(de klasse classification tree) zijn er de volgende mogelijkheden:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Entities** | **Doel** | **Toegelaten parents** | **Toegelaten**  **Afstammelingen** | **Test case toepasbaar** |
| Root | -denotes the test object |  | \*compositions  \*classifications | x |
| Compositions | -consist-of relationship  -to model an aspect that is broken down into other aspects  -to refine a value consisting of several parts | \*root  \*composition  \*class | \*compositions  \*classifications | x |
| Classifications | -is-a relationship  -to model an aspect that is partitioned into a collectively exhaustive and mutually exclusive set of values (equivalence classes) | \*root  \*composition  \*class | \*class | x |
| Classes | -to model equivalence classes (set of values that elicits the same behavior from the TO) and actual test values | \*classification | \*composition  \*classification | V |

1. Het moet mogelijk zijn om elke entity een andere naam of uitleg te geven. Eigenschap properties
2. Vertrekkende vanuit de classification tree moet het mogelijk zijn om test cases te laten genereren die voldoen aan de eisen van de gebruiker. De ingebouwde functies voor test cases zijn:

* Minimal combination creates a test suite that uses every class from each classiﬁcation at least once in a test case.
* Pairwise combination creates a test suite that uses every class pair from disjunctive classiﬁcations at least once in a test case
* Complete combination creates a test suite that uses every possible combination of classes from disjunctive classiﬁcation in a test case

1. De testcases moeten ook handmatig opgesteld kunnen worden aan de hand van een simpele query. Deze query moet de mogelijkheid geven om alle objecten aan elkaar te koppelen via een \* of een + teken.
2. De tool moet alle testcases die gemaakt worden voor een classification tree overzichtelijk en visueel kunnen weergeven.
3. De testcases worden onderverdeeld in verschillende groepen. Bijvoorbeeld al de groepen die gedefinieerd zijn bij ingebouwde functies.
4. Één testcase of een hele groep testcases kan aangevinkt worden als voltooid.

# Design functionaliteiten

1. De applicatie moet een duidelijk visueel beeld geven van de classification tree met daarop volgend de test cases.
2. Het moet mogelijk zijn om de classification tree een auto layout te geven zodat alles terug op zijn plaats komt.
3. De applicatie is overkoepelend, hierdoor kunnen er meerdere files geopend worden in hetzelfde programma.
4. De applicatie is voorzien van een werkbalk waar snelle acties mee uitgevoerd kunnen worden. De acties:

* New file
* Open file
* Save file
* Undo last
* Redo last
* Auto layout
* Browse forward to other file
* Browse backward to other file

1. De mogelijkheid moet bestaan om de layout voor de classification tree aan te passen (tekst, kleur, grote). Om deze aantrekkelijker te maken. Origineel zijn ze allemaal gelijk.
2. Het moet simpel zijn om elke entiteit te slepen, verplaatsen of verwijderen via de interface.

# Performance

1. De applicatie moet de file openen in 5 seconden.
2. De applicatie mag niet crashen bij het uitvoeren van eendere welke actie.

# Usability

1. Wanneer de applicatie zijn vorm begint te krijgen en de hoofdfunctionaliteiten aanwezig zijn, zal de applicatie getest worden door 5 of meer personen. Zij zullen enkele basis acties uitvoeren, als deze basis acties binnen aanvaardbare tijd voltooid worden en er geen moeilijkheden ondervonden worden zal de applicatie volstaan aan de eisen van usability.
2. Na de tests zal de vraag gesteld worden om de UI te beoordelen op uitzicht en bruikbaarheid. Beiden met een cijfer op 10.

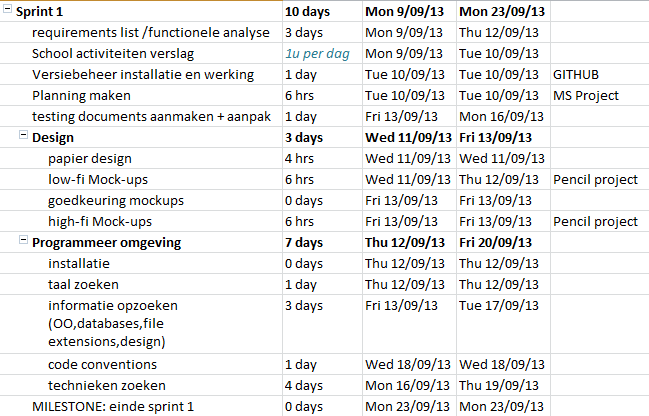
# Optionele functionaliteiten

*Deze functionaliteiten zullen enkel geïmplementeerd worden als er nog tijd is en de andere functionaliteiten die in dit document beschreven worden klaar zijn.*

1. Het moet mogelijk zijn om de testcases te exporteren naar Excel
2. Het moet mogelijk zijn om de gemaakte applicatie te implementeren in andere test tools
3. De tool moet de mogelijkheid hebben om een resultaat op te geven en deze te testen aan de hand van een IF-functie. Deze IF functie moet elke testcase afgaan (de mogelijkheid implementeren om enkele een bepaalde groep van testcases te doen).
4. De applicatie moet de mogelijkheid geven om requirements in te stellen op classes. Dit aan de hand van een simpele query. Bijvoorbeeld: instellen dat iemand van 14 jaar niet op pensioen kan zijn. Met deze requirements moet rekening gehouden worden wanneer de test cases gegenereerd worden.
5. De applicatie moet de mogelijkheid geven bij het aanmaken van een file om onmiddellijk alles te definiëren. Dit door middel van een venster waarin ingesteld kan worden: de root, de compositions, de classifications en de classes.
6. De functie om de CT (classification tree) aan te passen op design vlak.

# Scope (sprints scrum)

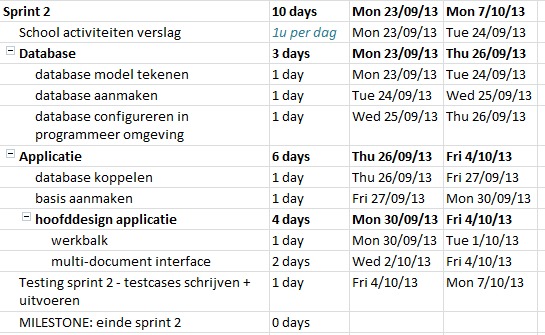
## Sprint 1



1. Toelichting sprint 1:

Deze sprint bestaat vooral met het opzetten van de basis. De mock-ups zullen hier voorgesteld worden aan de opdrachtgevers en mits hun goedkeuring verder uitgewerkt worden in high fidelity mock-ups die de basis zullen tonen van de applicatie.   
  
De 2de week van sprint 1 zal vooral in teken staan van het opzoeken en opfrissen van de gekozen programmeertaal/omgeving. Ook zullen de code conventions in deze week bepaald worden, deze zullen nuttig zijn bij het doorgeven van de applicatie.

## Sprint 2



1. Toelichting sprint 2:

Sprint 2 staat in het teken van de database. In deze database zal alle informatie bijgehouden worden over de gemaakte testcases en de classification tree indeling.   
  
Wanneer de database aangemaakt en geïmplementeerd is binnen de programmeeromgeving zal er aan de basis van de applicatie gebouwd worden namelijk de hoofdinterface (overkoepelende interface).

## Sprint 3



1. Toelichting sprint 3:

Sprint 3 = Classification tree.

In de eerste week van sprint 3 wordt de basis gelegd voor het aanmaken van de CT (classification tree). De klasse zal ontworpen worden en vervolgens geprogrammeerd worden. Wanneer de klasse correct werkt zal er voor gezorgd worden dat de elementen visueel aangemaakt kunnen worden in de file.   
  
Het uitvoeren van de nodige tests is in deze sprint zeer belangrijk aangezien de classification tree de basis is voor het genereren van de testcases. Als dit niet naar toebehoren werkt zal er niet aan sprint 4 gewerkt kunnen worden.

## Sprint 4



1. Toelichting sprint 4

Sprint 4 = testcases gebaseerd op classification tree.

Bij de start van sprint 4 zullen eerst de ingebouwde functies geprogrammeerd worden. Vervolgens de mogelijkheid programmeren dat de gebruikers ook zelf een functie kunnen aanmaken die testcases genereerd.   
  
Wanneer de testcases succesvol aangemaakt worden kan de programmatie beginnen om ze visueel te tonen.

## Sprint 5 en 6



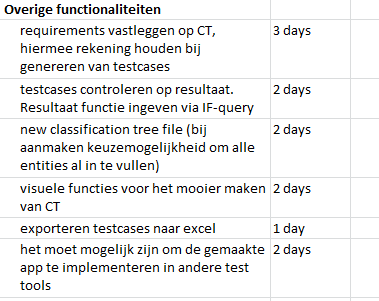
1. Toelichting sprint 5

Wanneer sprint 5 van start gaat zouden alle hoofdfunctionaliteiten van de applicatie aanwezig moeten zijn. Bij sprint 5 is het de bedoeling om de laatste aanpassingen te doen en enkele kleine bugs uit de applicatie te halen.

1. Toelichting sprint 6

Sprint 6 is de kleinste sprint, deze duurt namelijk maar 6 dagen en zal met doel hebben de applicatie achterlaten zodat deze later nog uitgebreid kan worden. Ook zal de presentatie over deze applicatie en de documentenstroom voor de PXL in deze sprint verder afgewerkt worden.

## Overige functionaliteiten



1. Toelichting functionaliteiten

Deze functionaliteiten zullen enkel gemaakt worden als er ruimte vrij komt in de planning. Met andere woorden als een geplande functionaliteit sneller af is als gepland. De functionaliteit die gemaakt zal worden als er tijd over is zal op dat moment besproken worden met de opdrachtgevers.