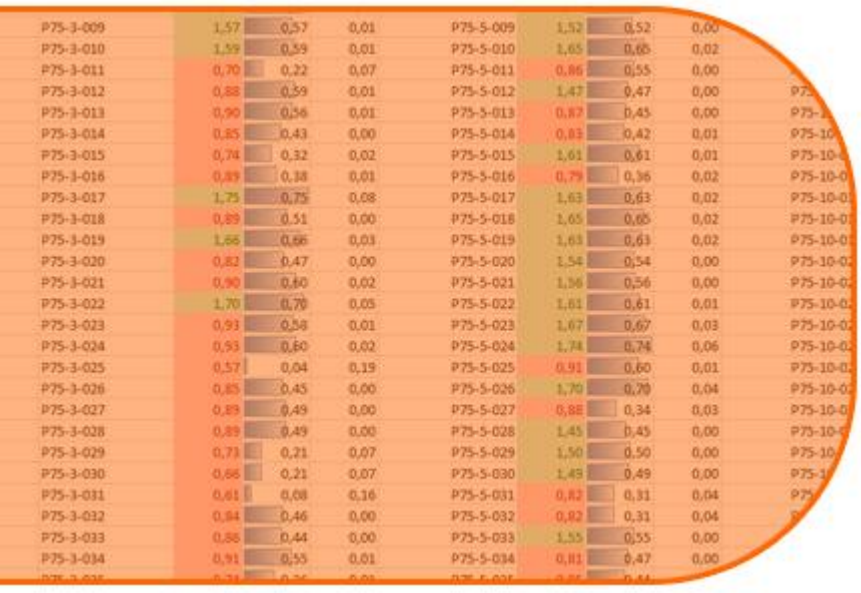
Visualisatiemethoden voor Multidimensionale Vergelijkingen

Bijlagen



Tijs van der Horst, 23 aug 2011

Opdrachtgever:
H. Tragter

Begeleider:
H. Tragter

2e lid Examencommissie:
Prof. W. Poelman

UT/IO 2011-04/2011-08
Datum: 23 augustus 2011

Aantal: 3 + digitaal
Aantal Pagina's: 19

Postadres:
Universiteit Twente
Industrieel Ontwerpen
t.a.v. de heer H. Tragter
Gebouw Horst, kamer W.262
Postbus 217
7500 AE Enschede

Inhoudsopgave

1. Interview Software Engineer/Expert CSS

Naam Geïnterviewde:	Hans Tragter
Naam Interviewer:	Tijs van der Horst
Datum:	23 maart 2011

Het CSS

Wat is zijn de doelen van het Computational Synthesis System (CSS)?

De doelen van het CSS zijn het verkorten van ontwerptijden, het verbeteren van de kwaliteit en het vastleggen van ontwerpkennis in het systeem. De eerste twee doelen worden bereikt door het mogelijk genereren van een grote hoeveelheid oplossingen. Dit kan met behulp van een computer snel en bovendien zit er bij een grote hoeveelheid oplossingen voor de gebruiker altijd wel een geschikte tussen.

Bij het maken van het systeem wordt kennis over het ontwerpprobleem vastgelegd zodat het programma deze kan toepassen voor berekening van oplossingen. Daarnaast wordt kennis altijd ter beschikking gesteld, ook wanneer de specialist ziek is of wanneer het als handboek wordt gebruikt.

Bij wat voor type ontwerpproblemen kan het CSS niet gebruikt worden?

Het CSS kan alleen gebruikt worden bij ontwerpproblemen waar parameters beschikbaar zijn die berekend kunnen worden.

Uit welke modulen bestaat het systeem/de software?

Het systeem is opgebouwd uit ruwweg vier onderdelen: parameters, kennis, het genereren van oplossingen en de GUI. Parameters worden beschreven in Integers, Floating Points of Booleans. Kennis is een set regels dat het ontwerpprobleem omschrijft, anders gezegd: een set met regels waardoor oplossingen te berekenen zijn. Dit gebeurd bij het ‘genereren van oplossingen’. Door middel van de GUI worden oplossingen aan de gebruiker gepresenteerd.

Wat wordt er gemaakt bij een nieuw ontwerpprobleem, en wat blijft hetzelfde?

Bij het maken van een CSS systeem moeten Parameters en Kennis volledig opnieuw samengesteld worden. De GUI en de algoritmen voor het maken van oplossingen blijven hetzelfde.

Wat is de invoer en uitvoer van het systeem ten opzichte van de GUI?

De Graphical User Interface is in principe dom en staat los van de rest van het systeem. De GUI krijgt van het CSS een set oplossingen aangeleverd die bestaan uit een groep bij elkaar horende waarden van parameters. De GUI levert het gelieve aantal oplossingen en een set met parameters (met vaste waarden, ranges of zonder waarden), deze worden door het CSS gecontroleerd en oplossingen worden gegenereerd.

Worden variabelen van te voren bepaald (specifiek bij het ontwerpprobleem), of wordt zoveel mogelijk open gelaten, zodat de persoon zelf kan bepalen welke variabelen vast staan en welke niet?

Variabelen kunnen worden ingedeeld in drie typen: Environment, Embodiment en Performance. Environmental variabelen zijn vooral vaste waarden (bv. temperatuur of druk) die in de kennis al zijn vastgesteld. Embodiment variabelen geven de range van het ontwerp aan, waarbij de gebruiker de minimale en/of maximale waarden vaststelt. De performance variabelen zullen zo veel mogelijk open gelaten worden. Dit zijn de variabelen waarop de gebruiker een keuze zal maken en geven de wens of prestaties van het ontwerp weer.

Zijn de oplossingen die het CSS genereerd altijd 0-dimensionaal?

Bij de oplossingen die het CSS genereerd hebben alle variabelen een vaste waarde, het is niet mogelijk om deze in ranges uit te drukken. Als een oplossing niet alle variabelen kan berekenen, dan is deze mislukt en zal ook niet worden weergegeven in de oplossing-set.

Hoe wordt een oplossing gegenereerd?

Het programma kiest (enkele) willekeurige waarden en berekend aan de hand daarvan de overige variabelen.

Zou het mogelijk zijn om de gebruiker enkele voorkeurswaarden in te laten vullen of eik-punten?

In één van de nieuwere versies van de GUI zit de mogelijkheid om een ‘set’ waarden voor een bepaalde parameter te geven, maar hiermee kan nog niet gerekend worden. Het zou een mogelijkheid kunnen zijn voor het systeem in de toekomst.

Huidige GUI

Welke programma’s, systemen en materialen heeft de gebruiker nodig om de software te gebruiken?

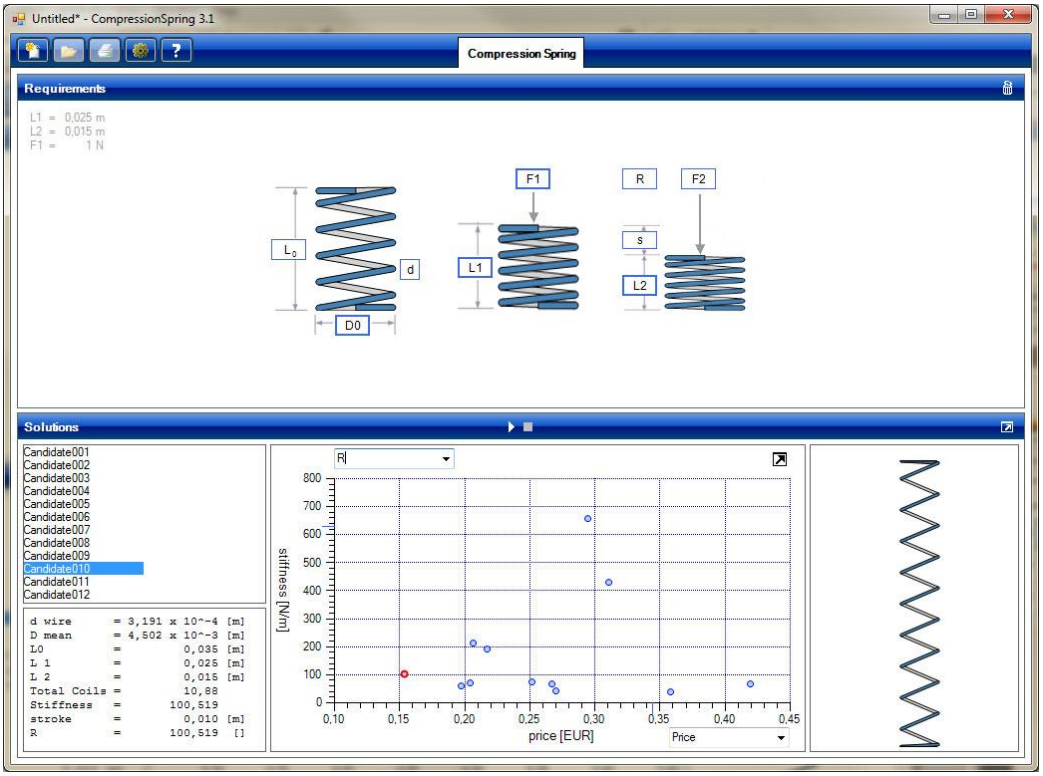
Met de huidige tests is het voldoende om een Personal Computer te gebruiken. De rekenkracht nodig voor de voorbeeld-ontwerpproblemen was niet heel groot, maar er was één ontwerpprobleem waarbij ten minste 4gb geheugen nodig was. Het is mogelijk dat er eisen aan de computer nodig zijn wanneer problemen complex zijn of in zeer grote getalen uitgevoerd moet worden, maar vooralsnog is daar weinig over bekend.

Welke handelingen moeten worden gedaan om de software te installeren?

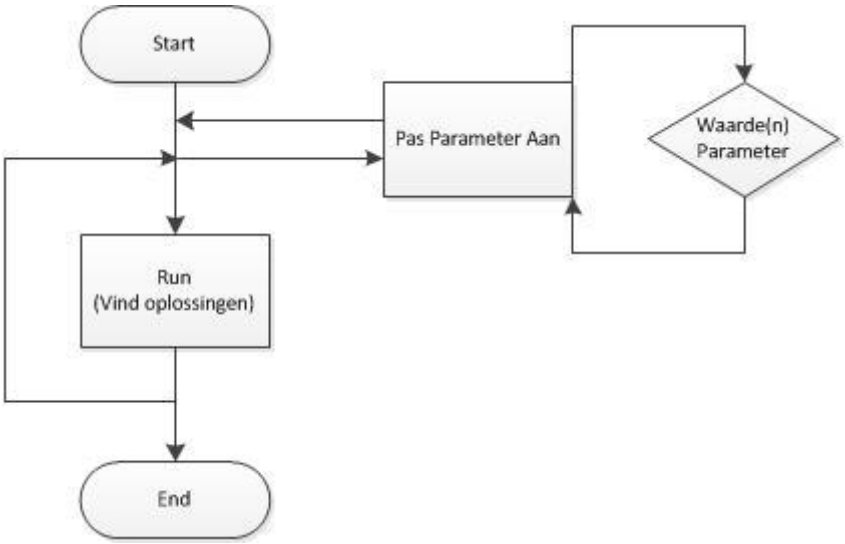
De huidige software kan zonder installatie direct gebruikt worden. Wel moet er gecommuniceerd worden met de klant om de nodige kennis en parameters in te voeren in het systeem. Er is nog geen editor beschikbaar waarbij gebruikers hun eigen ontwerpprobleem kunnen invoeren/programmeren, maar het is niet ondenkbaar dat dit in de toekomst zal kunnen.

Hoe start het programma op?

Het programma start op door middel van een executabel file. Deze communiceert met één andere file dat het ontwerpprobleem levert.



Figuur 1: Screenshot GUI



Figuur 2: Schematische tekening werking van de GUI

Welke handelingen moet de gebruiker verrichten om tot een oplossingsreeks te komen?

De GUI werkt eigenlijk heel simpel. Als het programma opstart zijn alle parameters en instellingen ingevuld. In principe staan alle parameters op 'vrij', tenzij kennis dit anders aangeeft. De gebruiker kan nu gelijk oplossingen zoeken, maar kan er ook voor kiezen om parameters aan te passen. Dit wordt gedaan door in het voorbeeldplaatje op deze parameter te klikken. Vervolgens krijgt de gebruiker de optie om lage en/of hoge limieten in te vullen óf een vaste waarde in te vullen.

Welke handelingen kan een gebruiker doen om tot een conclusie te komen?

De gebruiker kan de parameters aanpassen om zo tot een beter beeld van een (gedeeltelijke) oplossingsruimte te komen. De gebruiker kan verschillende waarden tegen elkaar uit zetten in een 2-dimensionale grafiek. Alleen twee parameters kunnen met elkaar vergeleken worden, maar een drop-down menu aan beide kanten van de grafiek kan gebruikt worden om elke mogelijke combinatie van parameters tegen elkaar uit te zetten.

Naast het vergelijken van verschillende parameters, kan de gebruiker er voor kiezen om een gedeelte van een oplossingsreeks te selecteren. Ook kan een gedeelte uitgesloten worden voor selectie, deze worden dan in het grijs aangegeven.

Welke overige functionaliteiten zijn aanwezig?

In de geteste GUI kunnen naast de standaard stappen ook nog instellingen gemaakt worden. Bij de instellingen kan de gebruiker aangeven hoeveel oplossingen het systeem moet proberen te genereren en hoeveel pogingen het systeem mag gebruiken om tot dat aantal te komen. Als het de CSS niet lukt om bij bepaalde variabelen een correcte waarde te vinden, gaat de CSS één variabele terug en berekend voor deze een nieuwe waarde. Het systeem kan maximaal drie keer een stap terug; als dan nog geen juiste waarde gevonden kan worden, is deze mislukt.

Gebruikers/Actoren

Wie maken het CSS?

In algemene zin zal de software engineer samen werken met de kennis technoloog. De kennis technoloog zal vervolgens weer in contact staan met een expert van het bedrijf/de instantie waarvoor het systeem ontworpen wordt.

Voor wie is het systeem bedoeld?

Hier is nog onduidelijkheid over. In principe zijn er veel mogelijkheden en bovendien zit het product nog in een fase waar deze aan veel klanten aangepast kan worden.

Het programma wordt waarschijnlijk gebruikt door een ontwerpafdeling. Voorbeelden zijn klanten die het product zelf willen gebruiken voor een routinematig ontwerpprobleem of een klant die een dergelijk complex ontwerpprobleem heeft dat dit onmogelijk is om met de hand te berekenen. Tevens zou het mogelijk zijn om deze tool online te zetten, zodat klanten van een bedrijf daar hun versie uit kunnen kiezen en deze vervolgens bestellen.

Wie gebruiken de software?

Bij het laatste voorbeeld zouden externen het programma gebruiken, in dit geval hebben gebruikers geen tot weinig voorkennis van het systeem. Daarnaast zou het als gereedschap gebruikt kunnen worden voor ontwerpers om zonder expert tot een oplossing te komen. Ook zouden experts het product kunnen gebruiken om meer gedetailleerd onderzoek te doen naar verschillende oplossingsmogelijkheden. Mogelijk zou zijn om enkel de kennis op te vragen dat in het product zit.

Wat is het doel van de gebruiker?

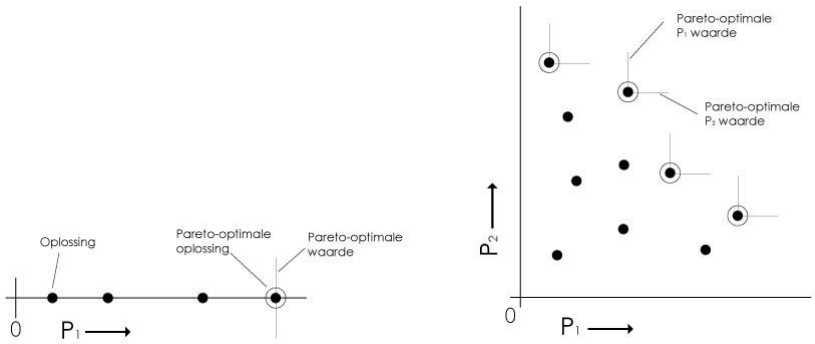
De gebruiker heeft altijd een ontwerpprobleem. De gebruiker zal op zoek gaan naar een ontwerp dat voor hem of haar het meest geschikt is. Dit hoeft dus niet een optimale oplossing te zijn.

Pareto-optimale oplossing

Aan welke eisen moet een oplossing doen om een 'Pareto-optimale oplossing te zijn?

De vraag wordt met een voorbeeld beantwoord. Stel dat er een set oplossingen wordt gegeven in een ééndimensionaal vlak. Dan is de beste oplossing degene die de hoogste parameter-waarde heeft. In figuur 3 (links) is dit weergegeven.

Stel nu dat er niet één, maar twee parameters zijn waarmee rekening gehouden moet worden. Nu is er geen één beste oplossing meer, maar een mogelijkheid van meerdere oplossingen. Het zou niet logisch zijn om een oplossing te kiezen dat slechter is in beide parameters dan een andere oplossing uit dezelfde set. Een oplossing waarbij geen alternatieve oplossing bestaat met een grotere of gelijke waarde voor elke parameter wordt ook wel een Pareto-optimale oplossing genoemd. In figuur 3 (rechts) zijn zulke oplossingen weergegeven waar gevraagd wordt naar een zo groot mogelijke parameter-waarde. Hier worden lijnen voor individuele waarden getekend om zo gemakkelijk de ruimte aan te geven waarbij geen andere oplossingen mogen zitten in dezelfde oplossings-set.



Figuur 3: Oplossingen in een ééndimensionaal (links) of tweedimensionaal (rechts) vlak.

Is er naast een Pareto-optimale oplossing niet ook één echte 'beste' oplossing?

Het zou mogelijk kunnen zijn om uit de Pareto-optimale oplossingen te bepalen welke wiskundig-technisch gezien een 'beste' oplossing is, maar hier zou de gebruiker niets aan hebben. Het is onmogelijk om alle factoren die van belang zijn door te geven aan het systeem, dus is het aan de gebruiker om een keuze te maken naar geschiktheid van een mogelijke oplossing.

Daarnaast is één van de doelen van het systeem om de gebruiker inzicht te geven in de oplossingsreeks: Het geven van één enkele oplossing laat de gebruiker niet weten of dit een geschikte oplossing is. Het is mogelijk om één enkele oplossing te krijgen uit het systeem, maar dit komt omdat alle overige berekeningen mislukt zijn. In een dergelijk geval zal de gebruiker moeten nadenken over de ranges die zijn ingesteld.

Welke ontwerpeigenschappen kunnen door de computer berekend worden en welke niet?

Het is onmogelijk om elke eigenschap aan te geven die niet door de computer berekend kan worden. Mogelijkheden zijn dat de gebruiker extra persoonlijke informatie heeft over productietijden en beschikbaarheid van technologieën om wel of niet voor een oplossing te kiezen.

Hoe worden oplossingen gerangschikt?

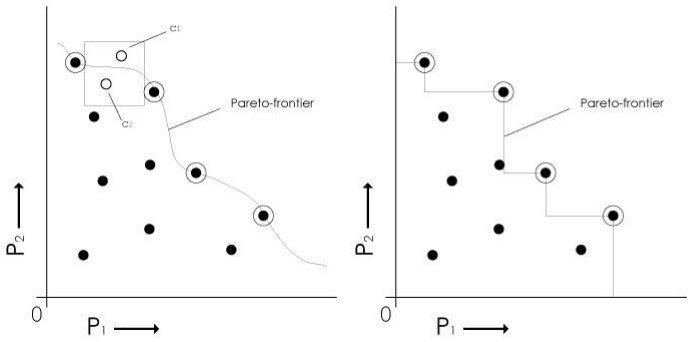
In het artikel 'How Synthesis Software Affects Engineering Design Tasks' (Tragter et al., 2010) wordt een Pareto-kwaliteit waarde aan oplossingen gegeven. Pareto-optimale waarden krijgen een genormaliseerde waarde 1 en overige oplossingen krijgen een waarde tussen 0 en 1, dat aangeeft hoe dicht deze bij een de Pareto-frontier staat. Een genormaliseerde waarde is gebruikt zodat erg uitstekende oplossingen niet de kwaliteitsvergelijking verstoren door deze uit te rekken. Deze Pareto-kwaliteit is alleen gebruikt ter vergelijking van de onderzoeksresultaten en zijn nooit laten zien aan de gebruikers. Dit is dus enkel een factor geweest dat resultaten zou kunnen vergelijken en aanduiden of deze correct waren en zo niet, in hoeverre niet.

Wat precies is een Pareto-frontier?

Een Pareto-frontier is een randfunctie dat aangeeft waar Pareto-optimale oplossingen (kunnen) zitten. Een effectieve Pareto-frontier kan alleen gemaakt worden wanneer alle variabelen op een continu vlak zitten. Zelfs dan moeten alle Pareto-optimale waarden op continuïteit berekend worden. Dit is onmogelijk en dus zijn er studies gedaan om een 'fictieve' Pareto-optimale lijn te maken.

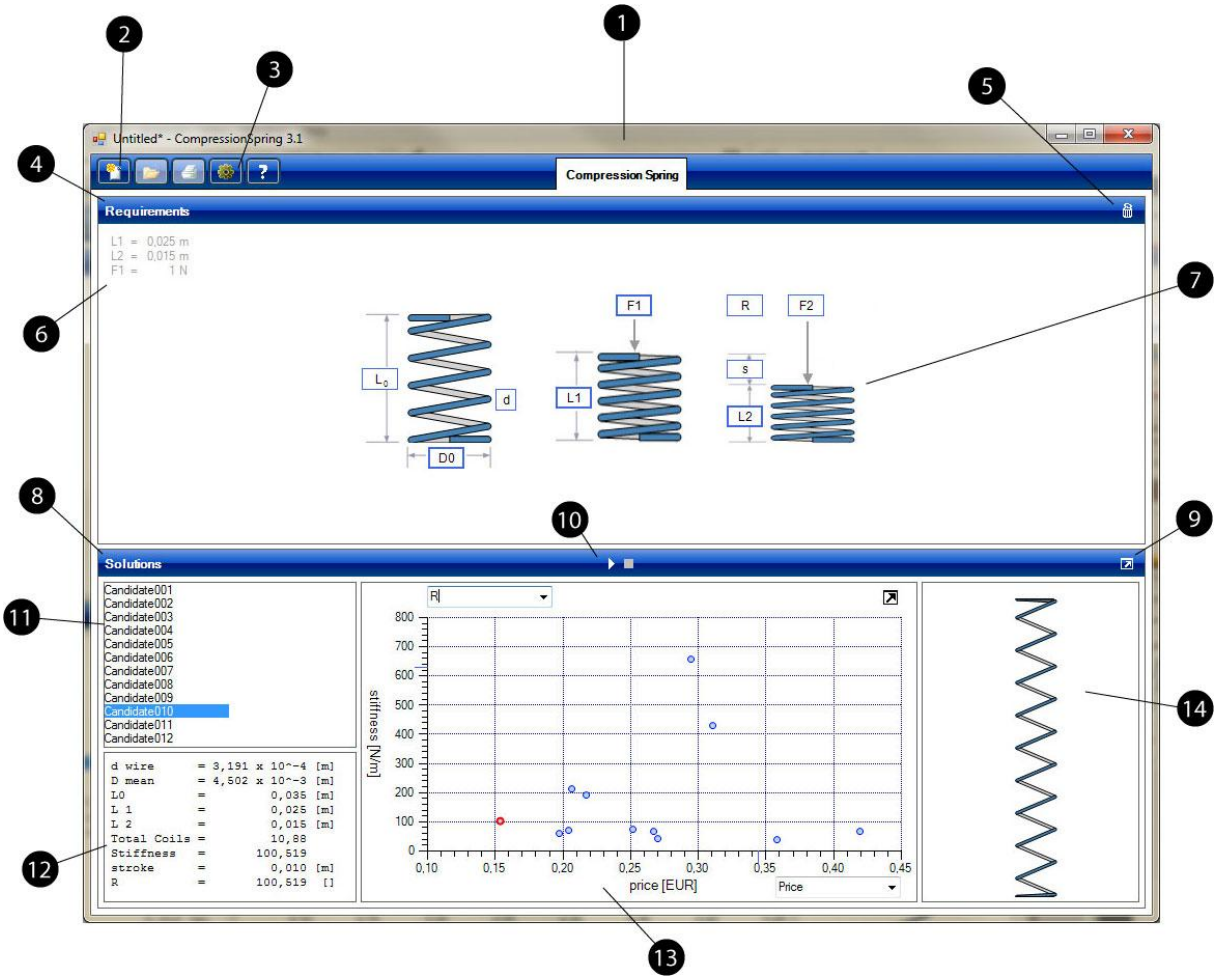
Het nadeel van deze lijn is dat het onmogelijk is om zeker te zeggen of deze wel op de werkelijke Pareto-frontier zit. In figuur 2.1 is aangegeven hoe een dergelijke lijn eruit kan zien. De kennis die klopt zijn de aangegeven Pareto-optimale punten. De lijn kan niet aangeven of er niet nog meer efficiënte punten zoals c1 zijn, of dat de lijn veel te optimistisch is en een werkelijk Pareto-optimaal punt bij c2 zou liggen.

Bij discrete variabelen is het helemaal niet mogelijk om een Pareto-frontier te creëren, want niet alle waarden kunnen bereikt worden. Voorbeeld van een discrete variabele is een bepaald materiaal (of de specificaties hiervan).



Figuur 4: Geschatte Pareto-frontier (links) en Zekere Pareto-frontier (rechts).

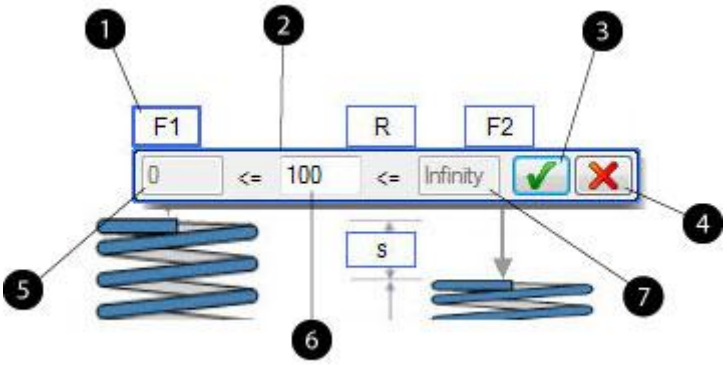
2. Beschrijving huidige GUI



Figuur 5: Grafische User Interface van het CSS systeem. Met deze versie is het gebruiksonderzoek in 2010 gedaan.

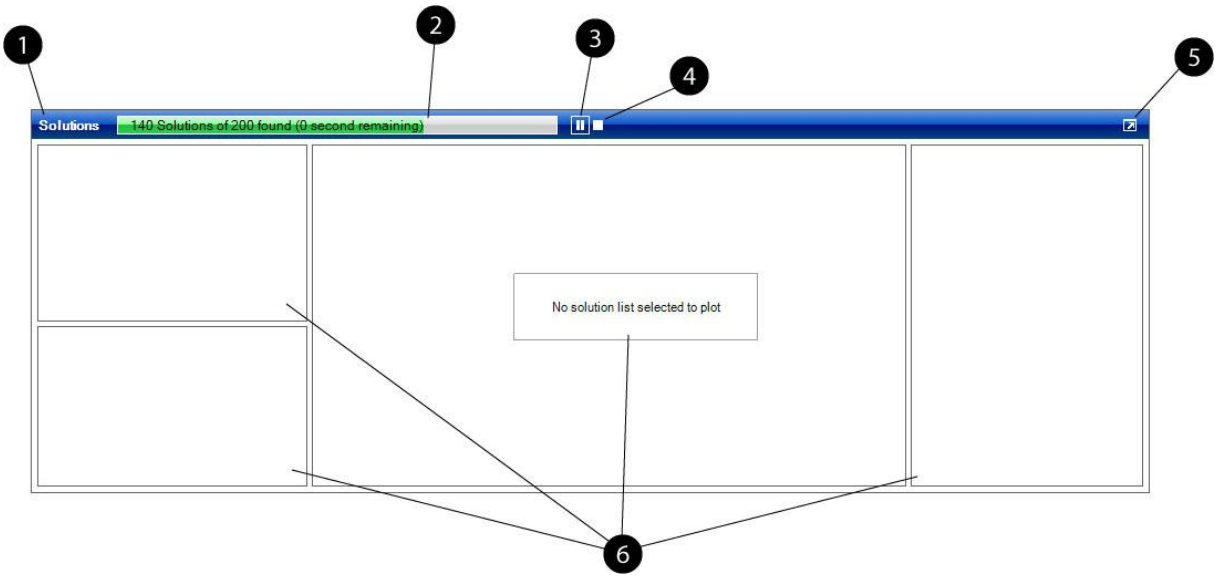
1) GUI Window van het CSS
2) De 'Nieuw' button: Alle gegevens worden verwijderd of op default gezet.
3) De 'Settings' button: Hier kunnen settings aangepast worden.
4) Eisen Frame: Gedeelte waar eisen van het ontwerpprobleem gezet kunnen worden.
5) 'Reset Requirements' button: Zet alle waarden van parameters op default.
6) Lijst van constante waarden (wordt geüpdate wanneer parameters aangepast worden).
7) Plaatje van het ontwerpprobleem, met bijbehorende parameters. De parameters kunnen worden geselecteerd.
8) Solutions window: Gedeelte waar oplossings-resultaten worden afgebeeld.
9) 'Resize' button: Als deze gedrukt wordt, neemt de Solutions window de volledige grote van de GUI aan. Als deze al als zodanig is, neemt deze de vorige afmetingen aan. De Solutions window kan ook handmatig aangepast worden door met de muis tussen de Solutions window en Requirements window te klikken en slepen.
10) 'Solve' button: Zodra deze wordt gedrukt wordt er met de ingestelde gegevens een oplossingsreeks gegenereerd.

11) Lijst met oplossingen. Geselecteerde oplossingen in de plot worden hier ook geselecteerd en visa versa.
12) Lijst met parameters en waarden hiervan. Wordt alleen laten zien als één enkele oplossing geselecteerd is.
13) 2-dimensionale plot van de oplossingen. Past zich automatisch aan het totale domein van de waarden van alle oplossingen.
14) Afbeelding van de oplossing. Wordt alleen laten zien als één enkele oplossing geselecteerd is.



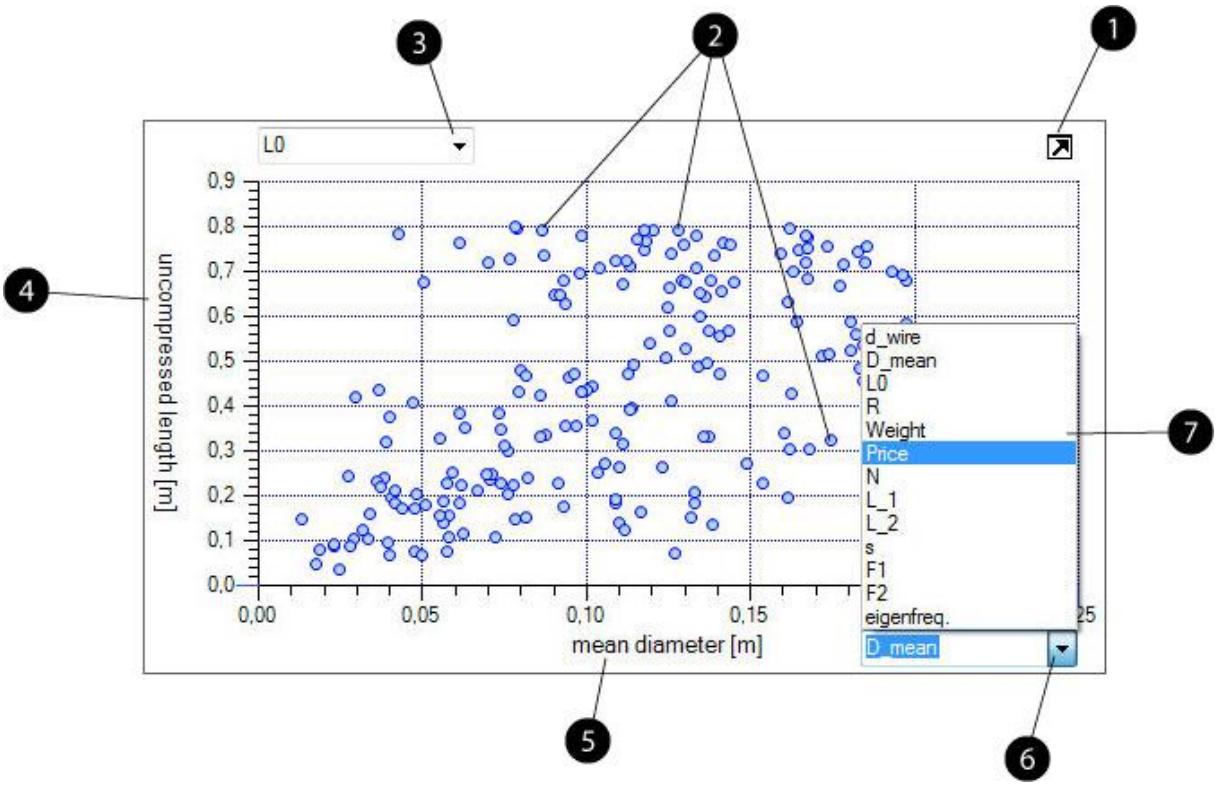
Figuur 6: Parameter-instellingen. Zo ziet het eruit als een parameter aangeklikt wordt in de GUI.

1) Parameter dat aangepast wordt.
2) Parameter-definieer-kader.
3) 'Opslaan en Exit'-button: Slaat de ingevulde waarden op en haalt het kader weg.
4) 'Niet-Opslaan en Exit'-button: Haalt het kader weg en behoudt de al opgeslagen waarden.
5) 'Minimale Waarde'-textfield: Hier kan de minimale waarde voor de parameter gezet worden.
6) 'Constante Waarde'-textfield: Hier kan een constante waarde voor de parameter gezet worden. Als deze is ingevuld, worden de minimale en/of maximale waarde niet gebruikt en grijs gemaakt.
7) 'Maximale Waarde'-textfield: Hier kan de maximale waarde voor de parameter gezet worden.



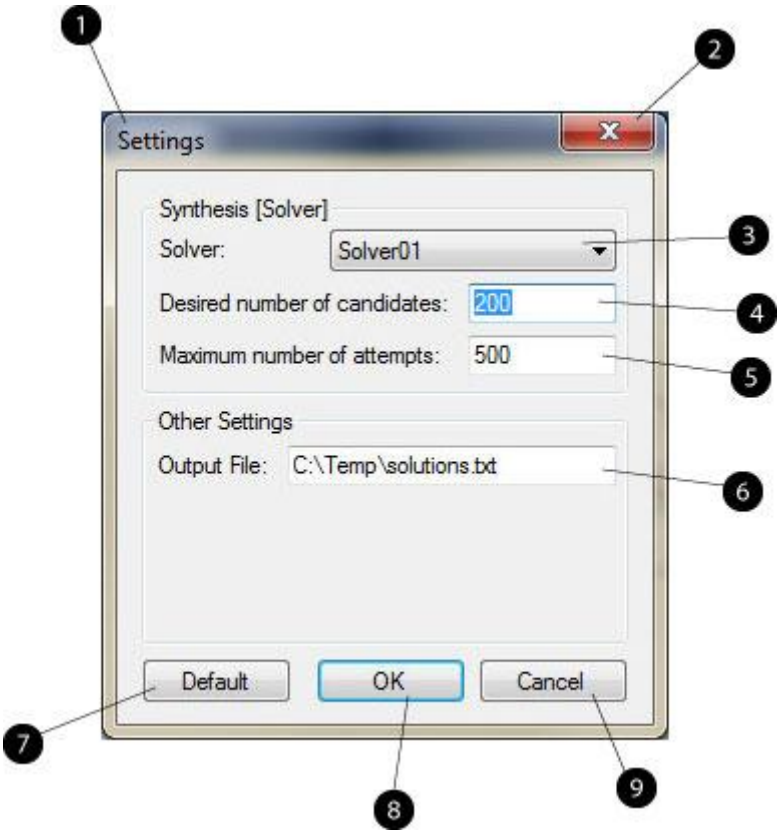
Figuur 7: Solutions-window, tijdens het genereren van oplossingen.

1) Oplossingen kader tijdens het genereren van oplossingen.
2) Balk dat aangeeft hoeveel oplossingen er zijn gevonden, hoeveel er zijn geprobeerd en hoe lang het ongeveer zal duren voordat alle pogingen gedaan zijn.
3) 'Pauze'-button: Wanneer ingedrukt zal het genereren van oplossingen gepauzeerd worden. De al gegenereerde oplossingen worden dan nog niet laten zien.
4) 'Stop'-button: Wanneer ingedrukt zal het genereren van oplossingen stopgezet worden. De al gegenereerde oplossingen worden weggegooid.
5) 'Resize'-button: Als deze gedrukt wordt, neemt de Solutions window de volledige grote van de GUI aan. Als deze al als zodanig is, neemt deze de vorige afmetingen aan. De Solutions window kan ook handmatig aangepast worden door met de muis tussen de Solutions window en Requirements window te klikken en slepen.
6) Lege kaders, als gevolg van het gebrek aan oplossingen.



Figuur 8: Plot van de oplossingsreeks.

1) 'Resize'-button: Als deze ingedrukt wordt, wordt er een nieuwe window gemaakt met daar in de plot. Meerdere windows kunnen gemaakt worden en hierbij onafhankelijk parameters ingesteld worden. De afmetingen van de windows kunnen ook naar gelang aangepast worden, evenals de positie.
2) Oplossingen, als een stipje rood gekleurd is (ipv het standaard blauw) is deze geselecteerd.
3) 'Parameter'-dropdownlist: Hier kan de parameter voor de verticale as ingesteld worden.
4) De verticale as, met bijbehorende informatie.
5) De horizontale as, met bijbehorende informatie.
6) 'Parameter'-dropdownlist: Hier kan de parameter voor de horizontale as ingesteld worden.
7) Lijst van parameters die geselecteerd kunnen worden.



Figuur 9: 'Settings'-window: Kader waarin algemene instellingen gemaakt kunnen worden betreffende het programma.

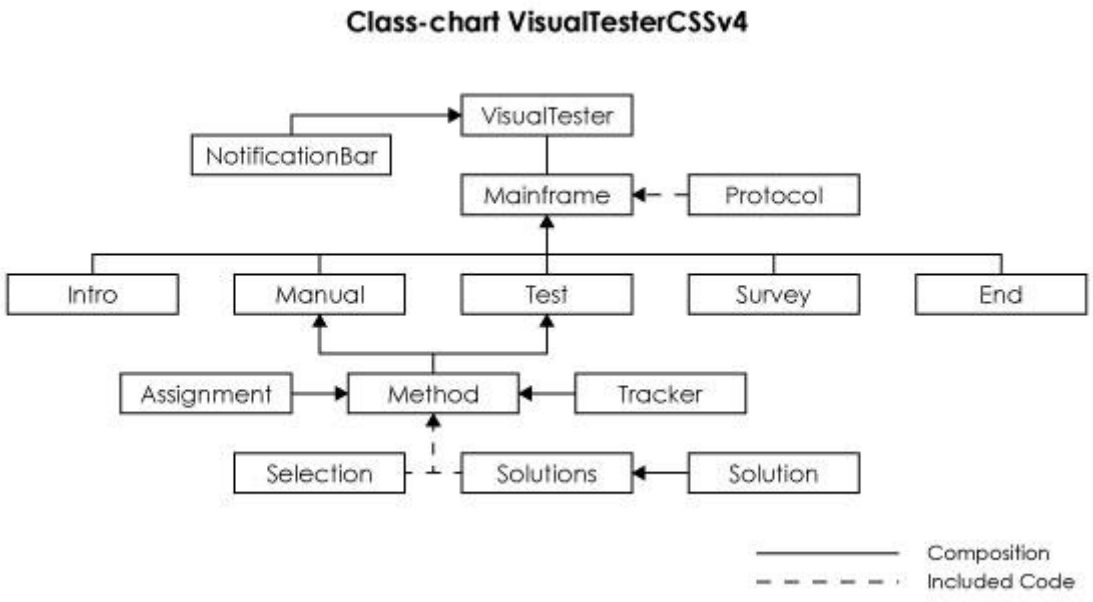
1) 'Settings'-window.
2) 'Exit'-button: Wanneer ingedrukt wordt het window weggehaald zonder dat instellingen opgeslagen worden.
3) 'Solver'-dropdownlist: Hier kan het algoritme geselecteerd worden dat gebruikt wordt om de oplossingen te genereren.
4) 'Desired Number of Candidates'-textfield: Hier kan het aantal gewenste oplossingen aangegeven worden.
5) 'Maximum Number of Attempts'-textfield: Hier kan het maximaal aantal pogingen tot het maken van een oplossing aangegeven worden (in het geval dat het gewenste aantal oplossingen niet gehaald wordt).
6) 'Output file'-textfield: Hier kan de URL gezet worden waar de oplossingen worden opgeslagen.
7) 'Default'-button: Zet alle instellingen terug naar de standaard-waarde.
8) 'Ok'-button: Slaat de waarden op en sluit het 'Settings'-window af.
9) 'Cancel'-button: Sluit het 'Settings'-window af zonder op te slaan.

3. Framework

Algemeen

De gebruikstest is gemaakt aan de hand van object georiënteerd programmeren. Dit betekend ruwweg dat de software is ingedeeld in blokken (klassen). Van deze klassen kunnen objecten worden gemaakt die ieder unieke informatie bevatten, maar voldoen aan de richtlijnen van deze klassen. De overkoepelende klasse van de gebruikstest is genaamd 'VisualTesterCSS'. Omdat het ontwerpen van de gebruikstest een dynamisch proces is geweest, zijn er meerdere versies ontstaan van het geheel. De uiteindelijke versie is versie 4, of "v4"; vandaar dat deze ook wel 'VisualTesterCSSv4' genoemd wordt. In de hoofdklasse is voornamelijk de uitlijning van verschillende onderdelen in de GUI bepaald. Zo staat er links een simpele tijdslijn, is er bovenin ruimte voor een opdrachtenbalk, onderin ruimte voor een notificatiebalk, rechts de 'verder'-knop en de overige ruimte kan benut worden door de hoofdframes, waarbij van te voren is bepaald dat de ruimte boven de 'verder'-knop gebruikt zal worden als legenda voor de methoden. De hoofdklasse is 1024 bij 640 pixels groot, waarvan 786 bij 600 pixels gereserveerd zijn voor het hoofdframe. Monitoren kunnen tegenwoordig veel grotere resoluties aan, maar browsers zijn vaak nog ingesteld op lagere standaard resolutie waarden, zoals 1024 bij 768. Daar gaan vaak ook nog eens de menu-ruimten vanaf. Om ervoor te zorgen dat gebruikers niet in de browser moeten scrollen om de gehele test te zien, is er gekozen voor de eerder genoemde resolutie.

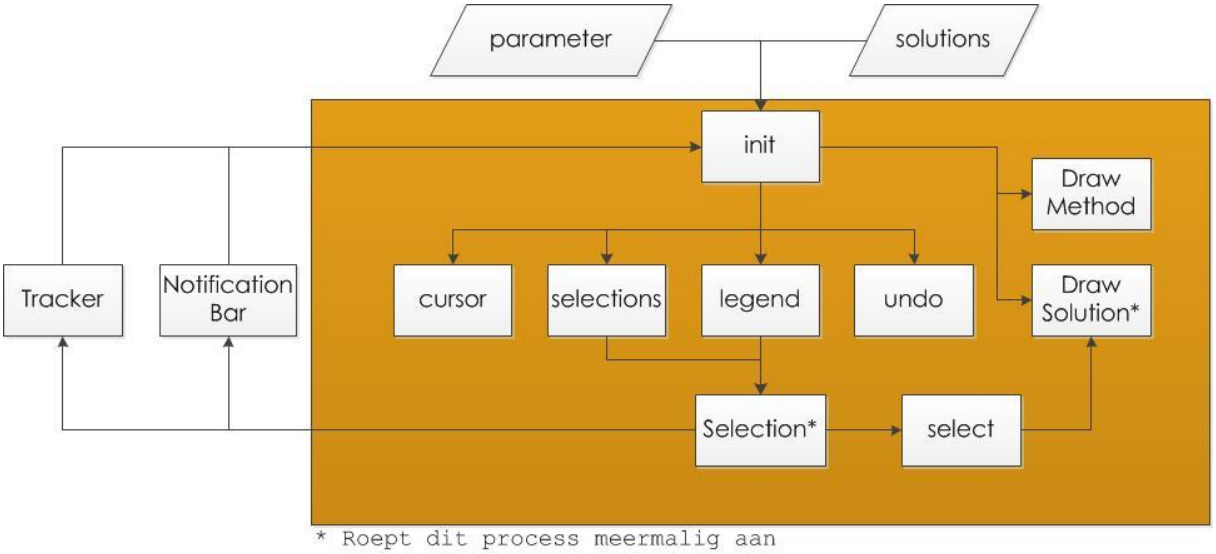
Er zijn vijf verschillende mainframes: Intro, Manual, Test, Survey en End. Elk hebben een eigen doeleinde en worden in deze volgorde door de gebruiker afgewerkt. Alleen bij de hoofdframes Manual en Test worden de methoden gebruikt. De methode communiceert vervolgens weer met andere klassen, zoals 'Assignment', 'Tracker', 'Selection' en 'Solutions'. De methoden en selectie gereedschappen zullen later in meer detail worden besproken. De Intro, Test en Survey frames communiceren ieder met de database. Ook de database zal later in meer detail worden besproken. In figuur 7 is een schematische weergave te zien van de het framework van de gebruikstest.



Figuur 10: Klassendiagram van VisualTesterCSSv4

Methode

Wanneer een methode object wordt aangemaakt, worden er enkele gegevens van buiten aangevraagd. Zo moet het object weten hoeveel parameters er worden gebruikt. Daarnaast wordt er aan de methode een set oplossingen gegeven (horende bij het aantal parameters). Ook wordt een 'tracker' object en de notificatiebalk doorgegeven, zodat de methode hier mee kan communiceren. In figuur 9 is een schematische tekening te zien van een methode klasse. Hierbij is het oranje vlak de klasse zelf. Processen of data die niet van de methode zijn, zijn buiten dit vlak weergegeven.



Figuur 11: Schematische weergave van de Methode klasse.

Wanneer het object wordt aangemaakt met de juiste instanties, wordt deze geïnitialiseerd[1] ('init') in onderstaande volgorde:

1. De methode wordt getekend.
2. Alle oplossingen worden getekend.
3. De cursor wordt klaargemaakt zodat deze veranderd wanneer de muis in het vlak van de methode komt.
4. De nodige selecties worden aangeroepen voor gebruik.
5. De legenda wordt aangemaakt.
6. De 'undo'-knop wordt aangemaakt.

Er moet worden aangemerkt dat een Selectie niet een aparte klasse is, maar een stuk code wat geïmporteerd wordt wanneer nodig; daarom dat deze binnen de methode is gezet. In principe zijn de uitwerkingen van de selectie en de legenda daarvan apart, maar met oog op clustering van informatie zijn deze in hetzelfde bestand gezet. Vandaar dat zowel het initialiseren van de selecties en de legenda naar een selectie wijzen.

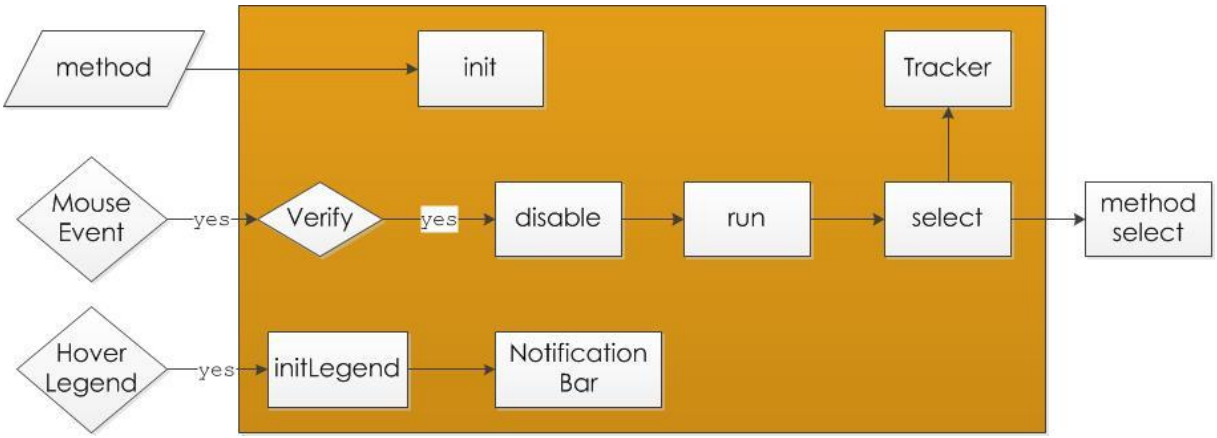
Wanneer de methode klaar is met initialiseren, wordt er gewacht op input van de gebruiker via de selecties. Wanneer een selectie gereedschap is gebruikt, wordt de nieuwe selectie doorgegeven naar de methode en deze tekent vervolgens de oplossingen opnieuw in de veranderde status.

[1] Initialiseren = Het proces van het omzetten van gegevens tot een object.

Dit is een algemene beschrijving van een methode, maar sommige methoden hebben unieke functies waardoor het framework iets is aangepast. Zo moet bij de multidimensionale methode ook de methode opnieuw worden getekend wanneer er geroteerd wordt en heeft de tabel een extra schuifbalk. Informatie over type methoden is beschreven in het hoofdstuk “Visualisatie Methoden”.

Gereedschappen

Binnen elke methode is aangegeven welke gereedschappen worden gebruikt. Niet alle gereedschappen zijn daadwerkelijk selectie gereedschappen. Het creëren van een selectie gereedschap is enkel het importeren van de code die nodig is voor de gereedschap. Alle gereedschappen hebben dezelfde opbouw, maar om overlap te voorkomen, hebben alle gereedschappen hun eigen ID gekregen dat voor alle functies en data gebruikt wordt. Een schematische weergave van een selectie gereedschap is weergegeven in figuur 10. Wanneer een gereedschap wordt gebruikt, wordt de initialisatie functie (‘init’) aangeroepen. Deze doet niets anders dan het activeren van zogeheten ‘luisteraars’, die controleren of een bepaalde actie is ondernomen door de gebruiker. Wanneer een herkenbare actie is gedaan (‘event’), wordt er gecontroleerd of deze bij het gereedschap hoort. Soms moet er nog een tweede actie gedaan worden om het gereedschap te activeren, dit wordt gecontroleerd bij ‘Verify’ (verifiëren). Wanneer een gereedschap is geverifieerd, worden alle andere gereedschappen gedeactiveerd om interferentie te voorkomen. Vervolgens wordt het gereedschap gestart en uitgevoerd. Wanneer de gebruiker het gereedschap afsluit (door de juiste handeling te verrichten), wordt de selectiefunctie van het gereedschap uitgevoerd, waarin gekeken wordt welke oplossingen geselecteerd moeten worden en welke niet. Deze informatie wordt vervolgens doorgegeven naar de methode. Tevens worden nu alle gereedschappen weer ingeschakeld en begint het proces opnieuw. Het legenda gedeelte van de gereedschappen is apart en het initialiseren hiervan gebeurd via de methode. De methode initialiseren de legenda’s van alle gereedschappen en stopt deze in een lijst. Deze lijst wordt vervolgens afgelopen bij het plaatsen van de legenda op de GUI. Wanneer de gebruiker over de legenda heen gaat, komt er een beschrijving van het gereedschap tevoorschijn. Figuur 10 geeft alleen een algemene schematische weergave van een selectie gereedschap. Niet alle gereedschappen hebben een verificatie nodig of maken gebruik van een proces waarbij andere gereedschappen gedeactiveerd moeten worden. Bovendien zijn er ook geen-selectie gereedschappen, die de methode niet op de selectie aanspreken, maar op een ander proces, zoals het opnieuw tekenen van de methode en oplossingen bij ‘roteren’. Omdat gereedschappen als pure code worden geïmporteerd, is het niet nodig om de structuur in de afbeelding te volgen, maar om logica in de code te houden is dit wel zo veel mogelijk gedaan.



Figuur 12: Schematische weergave van een Selectie Gereedschap deel-klasse.

Database

Het nadeel van Adobe Flash en Actionscript is dat deze software en programmeertaal gelimiteerde mogelijkheden hebben voor het communiceren met een database. Wel zijn er mogelijkheden om communicatie aan te leggen met een andere programmeertaal. Er is voor gekozen om een SQL database aan te maken en hiermee te communiceren via PHP bestanden. Elke keer als er informatie van de test moet worden opgeslagen, worden eerst de betreffende variabelen in het juiste formaat opgeslagen in de VisualTester, deze worden vervolgens doorgestuurd naar een PHP-script. De variabelen worden in het PHP-script opgepakt en doorgestuurd naar het SQL database. Dit lijkt omslachtig, maar de PHP-scripts zijn redelijk klein en omdat de gebruiker geen directe toegang heeft, wordt de data veilig doorgestuurd. Omdat het eenrichtingsverkeer is, is het nadeel dat er in de gebruikstest zelf niet wordt gecontroleerd of de data opgeslagen is. Er wordt er dus op vertrouwd dat het systeem goed in elkaar zit. Met deze voorkennis is dit goed gecontroleerd en getest.

Eén van de doelen is om te kijken wat het effect is van een toenemend aantal parameters. Dit kan onderzocht worden door de resultaten van de gebruiker bij eenzelfde methode en verschillende parameter-aantallen te vergelijken. Hiervoor is het belangrijk dat geïdentificeerd wordt welke gegevens bij welke gebruiker horen. Er moet een identiteit (id.) gemaakt worden, dat de gebruiker aan de resultaten kan linken. Dit wordt gedaan tijdens de introductie. De id. zal er als volgt uitzien: G_DDMMJJJJWWW, waarbij ‘G’ het Geslacht is, ‘DD’ de geboortedag, ‘MM’ de geboortemaand, ‘JJJJ’ het geboortejaar en ‘WWW’ de eerste drie letters van de woonplaats. De software controleert of de informatie die hiervoor is ingevoerd correct is en vult deze automatisch aan waar nodig zodat de id. altijd evenveel tekens bevat.

Er zijn drie verschillende typen tabellen in de database. Deze worden apart aangeroepen bij verschillende handelingen binnen de gebruikstest. Er wordt alleen geschreven naar tabellen en geen informatie opgehaald. De id. van de gebruiker blijft bewaard in Flash tijdens het maken van de gebruikstest. Wanneer de gebruikstest wordt afgesloten raakt deze verloren, maar hiervoor wordt de gebruiker gewaarschuwd. De drie tabellen zullen afzonderlijk besproken worden.

Tijdens de introductie moet de gebruiker persoonlijke gegevens invullen. Deze worden opgeslagen in de tabel ‘users’. De informatie is belangrijk voor het maken van de id. voor de gebruiker zoals zojuist beschreven is. Het beroep van de gebruiker wordt gevraagd om bij de resultaten verschillen te kunnen vergelijken tussen ontwerpers en overige beroepen. Wat ook opgeslagen wordt, maar de gebruiker niet hoeft in te vullen, is het IP van de gebruiker. Dit wordt gedaan om te herkennen wanneer één gebruiker de test veelvuldig maakt.

Vervolgens zijn er vier tabellen voor de testresultaten, één tabel voor elke methode. De tabellen hebben dezelfde structuur voor elke methode. De resultaten van het aantal parameters worden niet gescheiden in tabellen, maar wel erkent doordat deze informatie wordt opgeslagen in de tabel. Bij het opslaan van een testresultaat wordt de id. en het aantal parameters beide als ‘key’[2] gebruikt. Dit omdat elke gebruiker drie testresultaten in de tabel heeft staan na afloop van de gebruikstest, namelijk één resultaat voor elk parameter-aantal. Daarnaast worden per test de volgende onderdelen opgeslagen:

- De gekozen oplossing.
- De tijd van initialisatie tot afsluiting van een deeltest.
- Het aantal handelingen.
- Het aantal keer dat elk gereedschap is gebruikt.

De laatste tabel in de database wordt gebruikt om de antwoorden van de enquête op te slaan. Elke methode is afgekort naar een enkele letter. Deze letters worden op volgorde gezet van de gebruikers keuze voor elke vraag en als zodanig opgeslagen.

[2] Key = Een waarde of een combinatie van waarden dat een database-invoer uniek maakt.