Configuratietool

Applicatie Refactor voorstel

Bestand : RefactorVoorstel

Versie : 1.1

Datum van uitgifte : 14-6-2018

Opgesteld door : Koen Wartenberg

# Versiebeheer

|  |  |  |
| --- | --- | --- |
| **Versie** | **Datum** | **Beschrijving** |
| 0.1 | 16-4-2018 | Initiële versie |
| 1.0 | 18-4-2018 | Eerste opzet |
| 1.1 | 19-4-2018 | Document veranderd |

Inhoudsopgave

[1. Versiebeheer 2](#_Toc511898776)

[2. Inleiding 4](#_Toc511898777)

[3. Begrippen lijst 4](#_Toc511898778)

[4. Voorstellen 5](#_Toc511898779)

[4.1. Inleiding 5](#_Toc511898780)

[4.2. Route in de object tree 6](#_Toc511898781)

[4.2.1. Beschrijving 6](#_Toc511898782)

[4.2.2. Waar in de code 6](#_Toc511898783)

[4.2.3. Aanbeveling 6](#_Toc511898784)

[4.3. Vage Menu items 7](#_Toc511898785)

[4.3.1. Beschrijving 7](#_Toc511898786)

[4.3.2. Waar in de code 7](#_Toc511898787)

[4.3.3. Aanbeveling 7](#_Toc511898788)

[4.4. Parameter variabele fout 8](#_Toc511898789)

[4.4.1. Beschrijving 8](#_Toc511898790)

[4.4.2. Waar in de code 8](#_Toc511898791)

[4.4.3. Aanbeveling 8](#_Toc511898792)

[4.5. Save knop 9](#_Toc511898793)

[4.5.1. Beschrijving 9](#_Toc511898794)

[4.5.2. Waar in de code 9](#_Toc511898795)

[4.5.3. Aanbeveling 9](#_Toc511898796)

[5. Voorstellen waar niet meer aan begonnen gaat worden 10](#_Toc511898797)

[5.1. Fouten worden onjuist afgevangen 10](#_Toc511898798)

[5.1.1. Beschrijving 10](#_Toc511898799)

[5.1.2. Waar in de code 10](#_Toc511898800)

[5.1.3. Aanbeveling 10](#_Toc511898801)

[5.2. DAL heeft geen tot weinig structuur 11](#_Toc511898802)

[5.2.1. Beschrijving 11](#_Toc511898803)

[5.2.2. Waar in de code 11](#_Toc511898804)

[5.2.3. Aanbeveling 11](#_Toc511898805)

[5.3. Klassen zijn niet aan elkaar verbonden 12](#_Toc511898806)

[5.3.1. Beschrijving 12](#_Toc511898807)

[5.3.2. Waar in de code 12](#_Toc511898808)

[5.3.3. Aanbeveling 12](#_Toc511898809)

[5.4. Geen MVVM-patroon 13](#_Toc511898810)

[5.4.1. Beschrijving 13](#_Toc511898811)

[5.4.2. Waar in de code 13](#_Toc511898812)

[5.4.3. Aanbeveling 13](#_Toc511898813)

[5.5. Nutteloze XML 14](#_Toc511898814)

[5.5.1. Beschrijving 14](#_Toc511898815)

[5.5.2. Waar in de code 14](#_Toc511898816)

[5.5.3. Aanbeveling 14](#_Toc511898817)

[5.6. Business laag GUI-elementen 15](#_Toc511898818)

[5.6.1. Beschrijving 15](#_Toc511898819)

[5.6.2. Waar in de code 15](#_Toc511898820)

[5.6.3. Aanbeveling 15](#_Toc511898821)

[5.7. SQL-injectie is niet netjes 16](#_Toc511898822)

[5.7.1. Beschrijving 16](#_Toc511898823)

[5.7.2. Waar in de code 16](#_Toc511898824)

[5.7.3. Aanbeveling 16](#_Toc511898825)

[6. Samenvatting 17](#_Toc511898826)

# Inleiding

In dit document zal worden beschreven waar de meeste winst op behaald kan worden door refactoren in de applicatie en welke tijd hiervoor nodig is.

Onderaan het document zal een samenvatting/opsomming staan van alle punten. De recommandatie is hier ook bij vermeld.

# Begrippen lijst

|  |  |
| --- | --- |
| **Begrip** | **Beschrijving** |
| MVVM | Model-View-Viewmodel. Een patroon in WPF om GUI laag en Business laag volledig los van elkaar te kunnen bewerken. |
| WPF | Windows Presentation Foundation |
| MVP | Model-View-Presenter. Een patroon die gebruikt wordt in Windows forms applicaties en in de eerste WPF-applicaties. |

# Voorstellen

## Inleiding

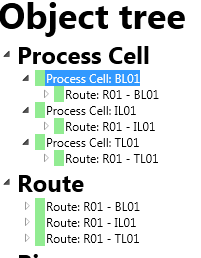
In dit hoofdstuk zal worden opgesomd waar knelpunten in de applicatie zitten en wat het voorstel is om deze op te lossen.

Er is ook geen documentatie over de structuur beschikbaar. Houd er dus rekening mee dat er op het moment weinig tot geen structuur in de applicatie zit. Verder moet je je telkens afvragen of de applicatie niet vanaf het begin opnieuw gebouwd had kunnen worden.

## Route in de object tree

### Beschrijving

Er staat een route kop in de object tree. Het doel hiervan is voor iedereen onduidelijk en kan dus beter weggehaald worden.



De aangemaakte routes staan al direct onder de processcell waar ze bij horen. Een extra tak voor deze objecten is vrij overbodig en maakt zaken alleen maar meer complex.

### Waar in de code

“MainWindow.xaml” en de achterliggende code die het ophaalt.

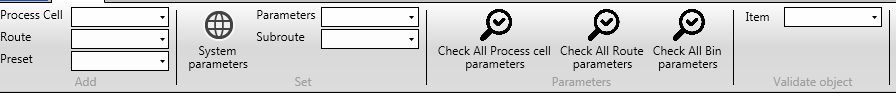
### Aanbeveling

De aparte route lijst is nutteloos dus het is goed voor het overzicht van de applicatie als die eruit gehaald wordt.

Dit is een redelijk eenvoudig om dit uit de applicatie te halen. Er is hier waarschijnlijk 1 dag voor nodig.

## Vage Menu items

### Beschrijving

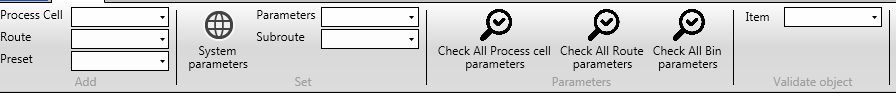


Het rood omringde deel van het start scherm is overbodig. Je kan met de bovenste twee uitklaplijstjes nieuwe routes of processcellen aanmaken. Je kan deze functionaliteit ook in de treeview terugvinden.

Het is verwarrend voor gebruikers om dezelfde functionaliteit op twee plekken aan te kunnen roepen.

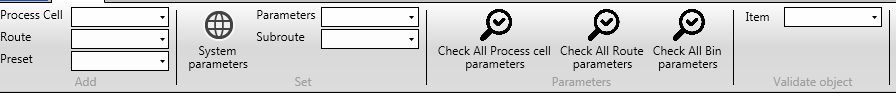
Fouten worden sneller ontdekt wanneer een functionaliteit, maar op een manier gedaan kan worden.

De “Preset” uitklaplijst is overbodig. Ten eerste, deze functionaliteit doet niets en voegen niets toe aan de applicatie. Ten tweede, soms wanneer je iets probeert te doen in de “Preset” uitklaplijst dan crasht de applicatie.



Deze twee uitklaplijstjes zijn ook schuldig van een dubbele functionaliteit.

Beide schermen die je kunt bereiken door op het juiste element in het lijstje te klikken zijn ook bereikbaar vanuit de treeview.



Dit uitklaplijstje is ook schuldig van een dubbele functionaliteit.

Je kunt uit deze lijst een object kiezen om te valideren. Dit kun je ook vanuit de treeview. In deze functionaliteit zit ook een fout. Wanneer je een object uit de lijst selecteert dan kan het zijn dat die alleen nog maar dat geselecteerde object controleert op validatie, tijdens de huidige runtime.

### Waar in de code

De menubalk op het beginscherm.

### Aanbeveling

Al deze functionaliteit is dubbel op. Het heeft geen toegevoegde waarde om deze functionaliteit erin te laten. Ook zitten in een paar van deze functionaliteiten fouten.

Het eruit halen van deze functionaliteiten bovengronds is binnen een halve dag gedaan. Als uit de hele applicatie alle methodes en de rest van de onderliggende code verwijderd moet gaan worden dan moet je rekening huiden met twee dagen

## Parameter variabele fout

### Beschrijving

Tijdens het uitzoeken van hoe je een “valid value” in moet voegen in de PAF-tabel is er gebleken dat deze in de applicatie helemaal niet op gecontroleerd wordt. Oftewel je kunt letters en min getallen invoeren bij een parameter waarbij de “valid value” 0 of meer moet zijn.



Valid value is groter of gelijk aan nul dat moet die zijn.





Dit kan gewoon worden ingevuld. Er worden namelijk niet eens foutmeldingen gegeven



Ja je kunt deze fout afvangen in de database, maar daar kan ik lastig bewerkingen in uit voeren. Laten we er dus even vanuit gaan dat dit in de applicatie moet gebeuren en voor zover het duidelijk is wordt hier helemaal niet op gecontroleerd.

### Waar in de code

Op het parameter scherm bij een parameter die geen uitklaplijst (dropdownlist) heeft. Hier wordt NIET gecontroleerd wat de “valid values” zijn.

### Aanbeveling

Deze functie moet gemaakt worden aangezien er in andere applicaties zeer veel stuk kan gaan doormiddel van deze slechte gegevens.

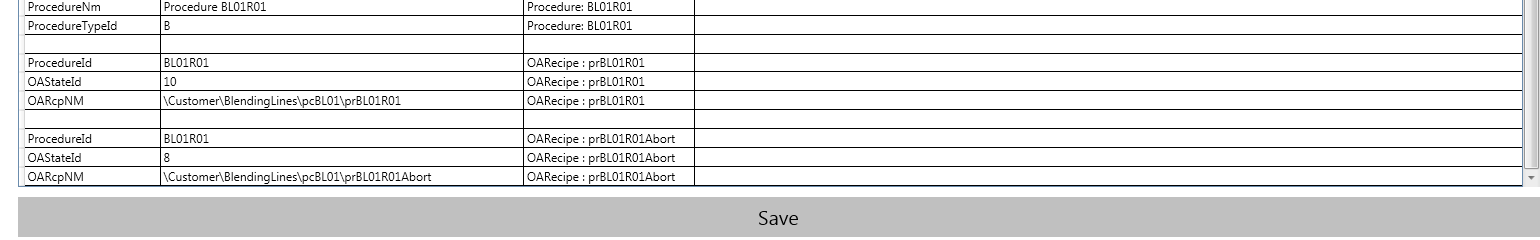
Een nette en correcte oplossing is bijna onmogelijk, doordat de code achter de “save” knop op hetzelfde scherm nogal rommelig is. Het kan waarschijnlijk opgelost worden, maar dan krijgt de code er wel een downgrade bij.

Naar schatting zal het al 5 dagen duren om deze fout in de huidige structuur eruit te halen.

## Save knop

### Beschrijving

Er staat een “Save” knop in de datagrid wanneer je hier iets in aanpast. Deze werkt volgens huidige bevindingen niet. Deze knop is dus overbodig en kan net zo goed worden weggehaald.



### Waar in de code

Op het hoofdscherm in de datagrid. De save knop verschijnt wanneer je iets in de datagrid veranderd.

### Aanbeveling

Het is aan te raden om de “Save” knop en de gerelateerde functionaliteit daaraan weg te halen. Om deze functionaliteit volledig te verwijderen kost 1 dag.

# Voorstellen waar niet meer aan begonnen gaat worden

## Fouten worden onjuist afgevangen

### Beschrijving

Fouten worden wel opgevangen, maar op plekken die meer uit gemakzucht zijn gedaan dan daadwerkelijk iets waar je wat aan hebt.

Wanneer er iets naar de database wordt toegeschreven verwijderde of veranderd dan gebeurd dit doormiddel van de bepaalde templates (methodes) in “BaseDataCommand.cs” (InsertTemplate, DeleteTemplate en GetTemplate).

Het probleem is alleen dat wanneer er iets fout gaat tijdens een bewerking op de database dan gaat het dus fout in die bepaalde methodes. En niet in de methodes waarvan het SQLCommand van afkomstig is, Daardoor kan het systeem de fout niet aangeven en weet je als programmeur niets. Je moet dan de fout zelf handmatig terug gaan zoeken en dit duurt lang aangezien alle bewerkingen van de database in een grote klasse staan van meer dan 3000 regels.

In de “WriteToDatabase” Methode Wordt een try catch gebruikt om te controleren of bepaalde insert statements niet werken. Het probleem hiervan is dat er een laag te hoog wordt gekeken na een potentiele fout oftewel de “try catch” hier is ook weer vrij nutteloos. Het systeem geeft hiermee het soort fout aan maar niet precies waar die zit.

### Waar in de code

“BaseDataCommand.cs” en ConfiguationController.cs (WriteToDatabaseMethode).

### Aanbeveling

De oplossing is niet zozeer moeilijk aangezien je er alleen voor met zorgen dat alle “try catches” op de juiste plaats moeten komen staan. Het probleem is alleen dat er heel veel methodes staan in een klasse van 3000 regels, dus deze bewerking vereist veel tijd met een grote kans op ombouw fouten.

Naar schatting duurt het 7 dagen om alles goed om te bouwen.

## DAL heeft geen tot weinig structuur

### Beschrijving

De gehele dal laag is opgebouwd uit 2 klassen, dit zijn “DatabaseController.cs” en “BaseDataCommand.cs”. Daarmee zit alle database logica in twee klassen. En beide klassen zijn in overtreding van de C# code guide Lines. Vooral “BaseDataCommand.cs” aangezien die gehele klasse meer dan 3000 regels telt.

Verder wordt “DatabaseController.cs” gebruikt als een soort abstractie laag. Het probleem hiervan is alleen dat deze abstractie laag niets toevoegt aan de leesbaarheid of structuur van de code. De klasse is eigenlijk gewoon nutteloos en kan net zo goed helemaal verwijderd worden.

### Waar in de code

De “DatabaseController.cs” en “BaseDataCommand.cs” klasse.

### Aanbeveling

Het ombouwen van de DAL laag is zo’n grote opgave dat dit op huidig moment bijna niet te doen is. Mijn advies is om gewoon een geheel nieuwe Data laag op te zetten aangezien de huidige niet meer te overzien is.

De schatting is dat dit 10 dagen gaat duren. Er moet eerst namelijk nog bedacht worden wat hiervoor de beste opzet is en hoe het dan in combinatie met de huidige structuur gedaan kan worden.

Persoonlijk raad ik het af om hieraan te beginnen aangezien je er meer hoofdpijn van zult krijgen dan resultaat.

## Klassen zijn niet aan elkaar verbonden

### Beschrijving

In de applicatie bestaan geen klassen zoals je ze normaal zou kennen. Op het moment worden de database tabellen direct vertaald naar C# klassen. Tussen deze klassen staan geen connecties doormiddel van objecten of lijsten met daarin objecten, maar door bijvoorbeeld een “string ID”. Dit zit dus super slecht in elkaar aangezien elke programmeur precies moet weten hoe hij moet controleren welk object heeft welke objecten.

Verder is er zeer weinig functionaliteit aan de klasse zelf overgelaten. Ook is veel functionaliteit aan de verkeerde lagen overgelaten. Hierdoor zijn stukken functionaliteit bijna niet terug te vinden.

### Waar in de code

De gehele map “DataClasses” behalve de map “Misc”

### Aanbeveling

Aangezien er over dit onderdeel geen documentatie van beschikbaar is en omdat de originele programmeur niet tot onze beschikking staat wordt er ten zeerste af geraden om dit deel van de applicatie te veranderen.

Dit deel van de applicatie staat geheel in contact met de data laag. Dit betekent dat als dit deel veranderd zou moeten worden dan de gehele data laag ook.

De tijd die het kost om te veranderen kan veel beter besteed worden aan het opnieuw opbouwen van dit deel van de applicatie. Oftewel het is beter om de gehele backend van de applicatie opnieuw te bouwen dan het te proberen te repareren.

Het aanpassen ervan zal veel meer tijd en moeite kosten dan het waard is. Verder zal het resultaat minimaal zijn in vergelijking met een geheel nieuwe constructie.

De schatting is dat deze bewerking 15+ dagen zal kosten.

## Geen MVVM-patroon

### Beschrijving

WPF wordt normaal gebruikt in combinatie met het MVVM-patroon. Het probleem is dat op het moment de applicatie is opgesteld tussen een mix van een paar WPF-technieken en het MVP-patroon.

Het probleem dat hierbij komt kijken is dat de applicatie hierdoor een stuk minder uitbreidbaar wordt en er kunnen hierdoor minder goede codeer regels opgesteld worden. Verder is de kans dat er fouten ontstaan tussen GUI-logica en Business logica een stuk groter dan met het MVVM-patroon.

### Waar in de code

Dit knelpunt zit in de gehele GUI laag en de eerste laag code daarachter. Zelfs contextmenu’s worden niet in de .xaml gezet, zoals het zou moeten.

### Aanbeveling

Om dit om te bouwen kost heel veel tijd. Je moet rekening houden dat bijna elk UI-scherm omgebouwd zal moeten worden (niet qua opmaak maar de manier waarop). In de applicatie zitten in totaal 13 verschillende schermen. Waarvan zo’n 4 a 5 veel aandacht nodig zullen hebben de andere zullen redelijk snel omgebouwd kunnen worden.

De schatting is dat alle schermen omgebouwd kunnen worden in 15+ dagen. Dit is een flinke klus die waarschijnlijk langer gaat duren dan verwacht, maar er is hier wel de meeste winst op te behalen.

## Nutteloze XML

### Beschrijving

Tijdens het ombouwen van de Parameterdefinities van de Xml naar de database is de ontdekking gedaan dat veel van de andere Xml sheets ook overbodig zijn. De helft van de Xml bestanden wordt gebruikt om een GUI te laten genereren. Dit is zelf in strijd met de normale MVVM-regels en is daarom zelf enorm slecht om te doen.

Verder worden in een deel van de Xml database wachtwoorden in opgeslagen, Hierbij vraag je eigenlijk om je gegevens gestolen te hebben.

### Waar in de code

Er is een zogenaamde “XMLController.cs” aangemaakt in de code. Via deze controller haalt de applicatie alle gegevens op uit de Xml bestanden.

### Aanbeveling

De applicatie moet van de Xml bestanden af totdat er echt noodzaak voor is (in dit geval is er al slecht gecodeerd). De helft van de Xml bestanden zijn GUI gerelateerd dit betekent dat ze doormiddel van het MVVM-patroon overbodig kunnen worden gemaakt.

Voor de andere Xml bestanden is er waarschijnlijk een kleine database aanpassing nodig.

De schatting is dat alle Xml bestanden overbodig kunnen worden gemaakt in circa 5 dagen. Het voordeel is dat alle data op een plek kan worden opgeslagen. Er moet voor de GUI dan wel een paar nieuwe schermen gebouwd worden

Of je zou gewoon een stukje functionaliteit er tijdelijk uit kunnen laten die toch niet echt nodig is. Zoals het ophalen van wachtwoorden of database connecties. Dit is niet essentieel voor een werkende applicatie.

De Schermen zullen nog wel omgebouwd moeten worden aangezien die op het moment op een slechte manier gegenereerd worden.

## Business laag GUI-elementen

### Beschrijving

In de applicatie is bijna geen gebruik gemaakt van de .xaml bestanden om een goede interface te maken. In plaats daarvan is besloten om de interface van bijna alle Windows te laten genereren door code. Dit maakt het bijzonder lastig om aanpassingen te doen, omdat een fout in de code ook een fout in de gui kan beteken. Het is hierdoor dus lastig om fouten op te sporen.

Dit hele verhaal komt eigenlijk neer op het feit dat een MVVM-patroon gebruikt moet gaan worden in de applicatie zodat fouten zoals deze bijna onmogelijk worden.

### Waar in de code

Dit knelpunt zit in de gehele GUI laag en de eerste laag code daarachter. Zelfs contextmenu’s worden niet in de .xaml gezet, zoals het zou moeten.

### Aanbeveling

(Dit is dezelfde aanbeveling als bij paragraaf 4.2.3. De tijd hiervoor staat gegeven is te verwaarlozen aangezien die aanbeveling eigenlijk dus 2 fouten kan oplossen)

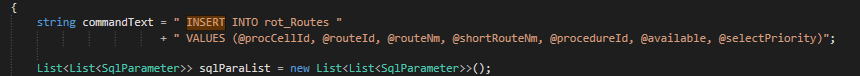
Om dit om te bouwen kost heel veel tijd. Je moet rekening houden dat bijna elk UI-scherm omgebouwd zal moeten worden (niet qua opmaak maar de manier waarop). In de applicatie zitten in totaal 13 verschillende schermen. Waarvan zo’n 4 a 5 veel aandacht nodig zullen hebben de andere zullen redelijk snel omgebouwd kunnen worden.

De schatting is dat alle schermen omgebouwd kunnen worden in 15+ dagen. Dit is een flinke klus die waarschijnlijk langer gaat duren dan verwacht, maar er is hier wel de meeste winst op te behalen.

## SQL-injectie is niet netjes

### Beschrijving

In de applicatie worden bijna alle insert statements gedaan op deze manier (de andere zijn al veranderd)



In dit statement worden niet de variabele gegeven die bij de ingegeven waardes horen. Dit kan twee problemen veroorzaken.

1. Het systeem voert het statement uit maar twee waardes zitten op de verkeerde plek. Hier is nu zeer moeilijk achter te komen of dit wordt veroorzaakt door het statement of iets anders in je code.
2. De applicatie kan stuk lopen op dit statement, maar omdat de variabele namen niet zijn aangegeven kun je niet controleren op welke die fout is gegaan.

### Waar in de code

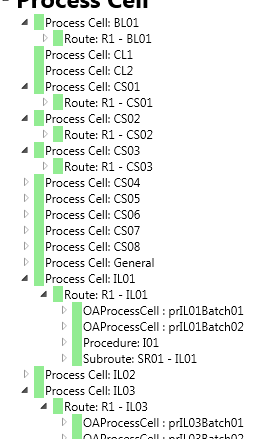
Dit een meerder stukje gelijke code staan in de “BaseDataCommand.cs”

### Aanbeveling

Dit probleem is redelijk makkelijk op te lossen. Het nut dat je hieruit haalt is vooral dat de code die je schrijft minder snel fout gaat en mocht dit zo zijn dan kun je snel zien waar dit aan ligt.

Het oplossen van dit probleem zal circa 3 dag kosten.

## TreeView klapt dicht naar meeste bewerkingen



### Beschrijving

De treeview klapt dicht wanneer de gebruiker bepaalde handelingen uitvoert. In zo’n 80 procent van alle bewerkingen klapt de treeview dicht. Dit heeft met bijna alle bewerkingen op de database te maken.

Het is voor de gebruiker enorm irritant als de treeview constant reset naar zijn ingeklapte staat wanneer je in de zelfde tak nog iets aan wilt passen

### Waar in de code

Op het hoofdscherm aan de linker kant.

In de applicatie dit het op meerdere plekken waarvan alle exacte locaties niet van achterhaald zijn.

### Aanbeveling

Dit probleem is bijna niet op te lossen in de huidige structuur van de applicatie. Het kan wel, maar dit zal wel veel tijd vereisen.

Naar schatting zal dit zo’n 15+ dagen vereisen. Dit komt doordat de applicatie slecht is opgesteld. Dan nog is dit wel een probleem om aan te gaan pakken, namelijk het gebruik van de applicatie zal hierdoor een stuk aangenamer zijn.

# Samenvatting

Hier staat een korte opsomming van de beschreven punten in een overzicht.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Paragraaf** | **Voorstelnaam** | **Korte beschrijving** | **Tijd (dagen)** | **Aan te raden** | **Prioriteit** |
| 4.2 | Route in de object tree | De Route tak in de objecttree is nutteloos en kan worden weggehaald | 1 | JA | High |
| 4.3 | Vage menu items | Er staan nutteloze en overbodige functionaliteiten in de menubalk op het hoofdscherm | 2 | JA | High |
| 4.4 | Parameter variabele fout | Er werkt gewoon een best belangrijke functionaliteit niet namelijk dat de valide waardes (voor een parameter) ook daadwerkelijk gecontroleerd worden of ze valide zijn. | 5 | JA | High |
| 4.5 | Save knop | Er staat een Save knop in de datagrid | 1 | JA | High |
| 5.1 | Fouten worden onjuist afgevangen | De “try catch” statements staan niet op de goede plaatsen. Hierdoor kan zeer lastig nagegaan worden waar een fout zit. | 7 | JA | Low |
| 5.2 | DAL heeft geen tot weinig structuur | De DAL heeft geen structuur maar wel een rare abstractie laag die eruit moet | 10 | JA | Medium |
| 5.3 | Klassen zijn niet aan elkaar verbonden | Er is helemaal geen klassenstructuur. Alle aanwezige klassen zijn eigenlijk direct vertaalde database tabellen.  (Opnieuw opbouwen is beter, het ombouwen hiervan is verspilde moeite met geen garantie op resultaat) | 15+ | NEE | Low |
| 5.4 | Geen MVVM-patroon | Er is geen design patroon gebruikt. In grotere applicaties is dit bijna altijd nodig (of zeer gewenst) | 15+ | JA | Low |
| 5.5 | Nutteloze Xml | De meeste Xml files zijn overbodig. De applicatie wordt hierdoor gecompliceerder dan nodig | 5 | JA | Medium |
| 5.6 | Business laag GUI-elementen | Er zitten veel Gui logica in de business laag en deze moet eruit aangezien dit erg de netheid en uitbreidbaarheid van de code aantast | 15+ | JA | Low |
| 5.7 | SQL-injectie is niet netjes | Insert commands zijn zonder de variabele name aangemaakt. Dit kan snel tot fouten lijden | 3 | JA | Medium |
| 5.8 | TreeView klapt dicht naar meeste bewerkingen | Na de meeste bewerkingen klapt dit treeview dicht. Dit is voor de gebruiker enorm irritant. | 15+ | NEE | HIGH |

\*De prioriteit is meer om aan te geven wat hoeveel nut de gebruiker heeft aan bepaalde bewerking. Daarna wordt pas gekeken na de overzichtelijkheid van de applicatie

Voorbeeld:

* Een toepassing binnen de structuur kan enorm handig zijn (JA), maar deze kan het misschien niet waard zijn om nu te gaan implementeren (LOW).