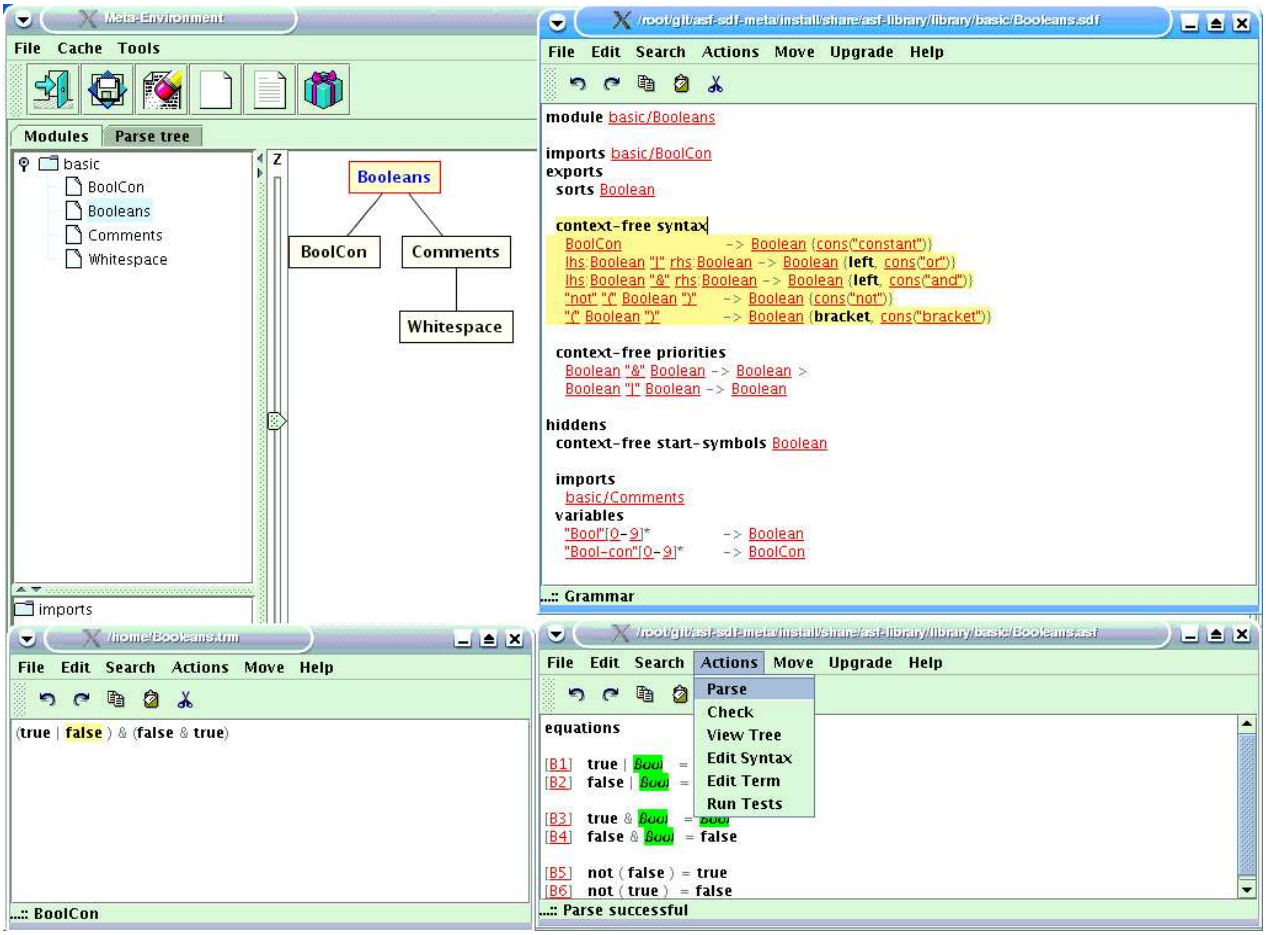
JFC/Swing Editor voor ASF+SDF Meta-Environment

S. Lakhloufi
November 2004
Afstudeerscriptie Masters Software Engineering



Opdrachtgever:
Centrum voor Wiskunde & Informatica
Kruislaan 413
1098 SJ Amsterdam
http://www.cwi.nl/
Afstudeerdocent:
Paul Klint
Stagebegeleiders:
Mark van den Brand, Hayco de Jong, Taeke Kooiker, Jurgen Vinju

Dankwoord

Het eind resultaat van mijn scriptie heb ik mede te danken aan een aantal mensen die mij door middel van ondersteuning en/of begeleiding hebben geholpen.
Op de eerste plaats wil ik Mark van den Brand, Hayco de Jong en Taeke Kooiker bedanken voor het mogelijk maken van van deze afstudeerstage. Zij hebben mij een uitermate geschikte opdracht gegeven die mijn technische kennis naar een hoger peil heeft gebracht. Ik heb van hun alle verantwoordelijkheid en vrijheid gekregen zodat ik dit project naar mijn eigen smaak kon voltooien. De ondersteuning die ik van hen heb gekregen heeft mij vaak weer de goede weg gewezen. Op de tweede plaats wil ik Jurgen Vinju bedanken voor een prettige samenwerking, zijn kennisoverdracht en zijn begrip voor grapjes.

Samenvatting

Software dient bruikbaar te zijn. De gebruikersinterface is datgene wat tussen mens en computer staat en is één van de belangrijkste factoren in de gebruikersvriendelijkheid van software. Gebruikersvriendelijkheid betekent dat het gebruik van de software overzichtelijk, eenvoudig en prettig overkomt. Met software die gebruikersvriendelijk is, worden irritaties en fouten in gebruik voorkomen. De gebruikers van deze software kunnen er sneller mee aan de slag.

De ASF+SDF Meta-Environment [1] is een interactieve ontwikkelomgeving en is ontwikkeld door het Centrum voor Wiskunde en Informatica (CWI) en de Universiteit van Amsterdam (UvA). ASF+SDF [1] staat voor Algebraic Specification Formalism + Syntax Definition Formalism en is speciaal geschikt voor het beschrijven van (programmeer)talen. Met ASF+SDF Meta-Environment kunnen taaldefinities gespecificeerd worden waaruit vervolgens tooling voor het manipuleren van de programma's over deze taaldefinities gegenereerd kan worden. Dit leidt tot automatisch gegenereerde, taalspecifieke omgevingen. De ASF+SDF Meta-Environment beschikt over een Grafische User Interface (GUI) die geschreven is in JFC/Swing.

Om ASF en SDF bestanden te kunnen maken of bewerken, maakt men gebruik van een aantal externe editors zoals Vim [10] en GNU Emacs [11]. Deze editors zijn geïmplementeerd in andere talen dan JFC/Swing waarin de GUI van de ASF+SDF Meta-Environment is geschreven. Deze editors zijn gekoppeld aan de Meta-Environment met behulp van de ToolBus [2] (zie paragraaf 2.2). Om de ASF+SDF Meta-Environment op platformen zoals Microsoft Windows(XP) en Mac OSX te kunnen draaien, moeten de juiste versies van deze externe editors beschikbaar zijn.

Tijdens mijn afstudeerstage heb ik mij bezig gehouden met het onderzoeken van de mogelijkheden van JFC/Swing om een editor voor ASF+SDF Meta-Environment te ontwikkelen. Een editor die geschreven is in JFC/Swing, is te integreren binnen de ASF+SDF Meta-Environment omdat de GUI van ASF+SDF Meta-Environment ook geschreven is in JFC/Swing. Daarnaast is een JFC/Swing Editor platformonafhankelijk. Dat houdt in dat de JFC/Swing Editor op elke computer kan draaien waarop Java Virtual Machine (JVM) beschikbaar is.

Het onderzoek heeft geresulteerd in een JFC/Swing Editor die gekoppeld is met de ASF+SDF Meta-Environment door gebruik te maken van de ToolBus. Deze JFC/Swing Editor biedt de functionaliteit die men kan gebruiken voor het bewerken van de ASF+SDF bestanden. De JFC/Swing Editor is ook voorzien van basisfunctionaliteit zoals Copy/Cut/Paste, Undo/Redo, Find/Replace en syntax highlighting. Via syntax highlighting kunnen de gebruikers de code veel sneller lezen aangezien de structuur van de code beter wordt weergegeven.

Inhoudsopgave

1 Inleiding

Voor u ligt de afstudeerscriptie van Said Lakhloufi. Dit is geschreven in het kader van mijn afstudeeropdracht voor de Masters Software Engineering aan de Universiteit van Amsterdam (UvA). De afstudeeropdracht heb ik gedurende een periode van tien weken uitgevoerd bij het Centrum voor Wiskunde en Informatica (CWI). Mijn taak was om onderzoek te verrichten naar de mogelijkheden die JFC/Swing biedt bij het ontwikkelen van een native editor.

Allereerst zal in hoofdstuk 2 de achtergrond en onderzoeksvragen besproken worden. In dit hoofdstuk worden tevens de probleemstelling, de opdracht, de verwachte resultaten en de tools & technieken die binnen het CWI worden gebruikt besproken.

Hoofdstuk 3 is een weergave van de JFC/Swing Editor. De nadruk wordt gelegd op de functionaliteit en de architectuur van de JFC/Swing Editor. Ook worden in dit hoofdstuk de gebruikte technologieën voor de realisatie van de JFC/Swing Editor aan de orde gesteld.

Vervolgens wordt in Hoofdstuk 4 de implementatie van syntax highlighting besproken. Hoofdstuk 5 bespreekt de bestaande tekst editors jEdit en Eclipse in het algemeen. Hoofdstuk 6 bevat de conclusies en de aanbevelingen. Als laatste de bibliografie en de bijlage.

2 Achtergrond en onderzoeksvraag

2.1 De ASF+SDF Meta-Environment

De ASF+SDF Meta-Environment[1] is een interactieve ontwikkelomgeving voor het automatisch genereren van interactieve systemen. ASF+SDF staat voor Algebraic Specification Formalism + Syntax Definition Formalism en is speciaal geschikt voor het beschrijven van (programmeer)talen. Een gebruiker voert een taaldefinitie in, daarbij geholpen en gecorrigeerd door het systeem. De opbouw van de generator is dusdanig dat direct met de afgeleide tools geëxperimenteerd kan worden. Enkele voorbeelden van tools die gegenereerd worden zijn:

- Parser;
- Type checker;
- Compiler.

Enkele voorbeelden van taaldefinities zijn:

- C;
- COBOL;
- Java.

De ASF+SDF Meta-Environment kan gebruikt worden in de volgende gevallen:

- U moet een formele specificatie voor één of ander probleem schrijven en u hebt interactieve steun voor dit proces nodig;
- U ontwikkelt uw eigen taal en wilt een interactieve omgeving voor deze taal tot stand brengen;
- U hebt programma's in één of andere bestaande programmeertaal en u wilt ze analyseren of omzetten naar een andere taal.

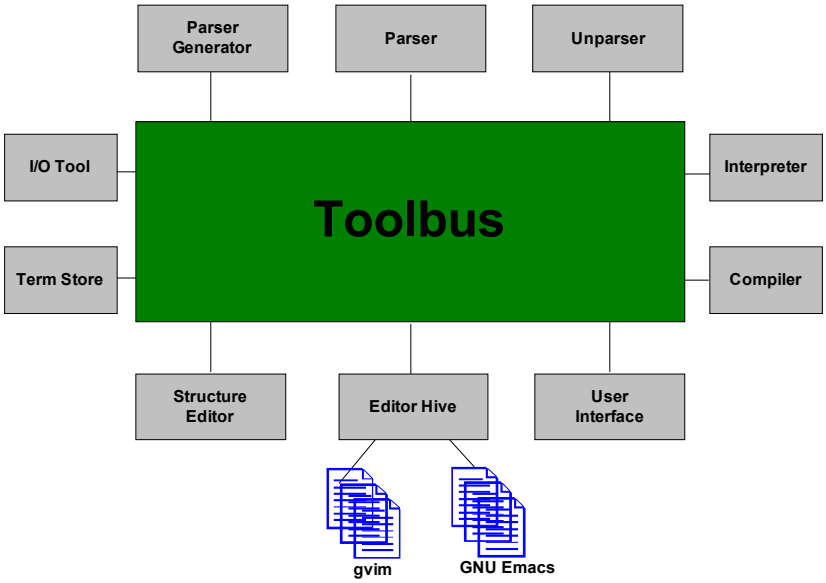
Door gebruik te maken van Meta-Environment kunnen nieuwe specificaties gemaakt en gewijzigd worden. Bestaande specificaties kunnen tevens getest worden. Een taalspecificatie bestaat uit onderdelen die bewerkt kunnen worden door gebruik te maken van de externe editors Vim [10] of GNU Emacs [11] .

2.2 ToolBus

ASF+SDF Meta-Environment zelf is een verzameling van een aantal componenten (tools) die met elkaar kunnen communiceren via de ToolBus [2]. ToolBus is ontwikkeld door de UvA en het CWI en is gebaseerd op proces algebra. Door gebruik te maken van ToolBus, kunnen componenten die in verschillende programmeertalen zijn geschreven aan elkaar gekoppeld worden. Een ToolBus script beschrijft een aantal processen die in staat zijn met elkaar te communiceren.

De ToolBus script wordt tevens gebruikt voor het genereren van de interfaces die voor de communicatie zorgen tussen de Meta-Environment en een van de componenten. De ToolBus interpreter creëert tifs bestanden die vervolgens door de tools Tifstojava en Tifstoc worden gebruikt om de interfaces te creëren. Tifstojava is de tool die interfaces creëert voor componenten die geschreven worden in Java. Om de interfaces te creëren voor componenten die geschreven worden in C wordt gebruik gemaakt van Tifstoc.

Figuur 1 laat de basisarchitectuur van de ASF+SDF Meta-Environment zien met de belangrijkste componenten.



Figuur 1: de basisarchitectuur van de ASF+SDF Meta-Environment

De ASF+SDF Meta-Environment beschikt over een GUI die geschreven is in JFC/Swing.

2.3 Probleemstelling

In de huidige situatie worden de externe editors Vim en GNU Emacs gebruikt voor het bewerken van ASF en SDF bestanden. Deze twee editors zijn in programmeertaal C ontwikkeld en de GUI van de Meta-Environment is ontwikkeld in JFC/Swing. Om de Meta-Environment op alle platformen te kunnen draaien, moeten voor elk platfom de juiste versies van deze editors beschikbaar zijn. De wens bestaat om een native editor te ontwikkelen in JFC/Swing. De gebruikers van de Meta-Environment kunnen dan kiezen tussen de native editor (JFC/Swing Editor) en de externe editors Vim en GNU Emacs.

Een editor die geschreven is in JFC/Swing is beter portabel. De broncode wordt gecompileerd in zogenaamde bytecode die uitgevoerd kan worden op elke computer waarop de Java Virtual Machine (JVM) is geïnstalleerd. De JVM zet de bytecode om naar code die uitgevoerd kan worden door de computer. Hierdoor is geen behoefte meer om platformspecifieke versies van de JFC/Swing Editor te ontwikkelen. Dat heeft als voordeel dat een versie van deze JFC/Swing Editor op elk platform waarop JVM is geïnstalleerd kan draaien. Als men in de toekomst wijzigingen wilt aanbrengen in de JFC/Swing Editor of deze editor van nieuwe functionaliteit wilt voorzien, dan hoeft men geen rekening te houden met elk platfom.

2.4 Opdrachtomschrijving

Mijn taak was om onderzoek te verrichten naar de mogelijkheden die JFC/Swing biedt bij het ontwikkelen van een editor. De JFC/Swing Editor moet beschikken over de basisfunctionaliteit voor tekstverwerker operaties en de functies die de Meta-Environment nodig heeft.

De basisfunctionaliteit is als volgt gedefinieerd:

- Copy/Cut/Paste om respectievelijk delen van de ingetikte code te kunnen kopiëren/verwijderen/plakken;
- Undo/Redo voor ongedaan maken van acties en heruitvoeren van die ongedaan gemaakte acties;
- Find/Replace voor het zoeken en vervangen van tekst. Deze functie is handig bij het corrigeren van typefouten.

De editors zijn afhankelijk van de input die zij krijgen van de Meta-Environment. De Meta-Environment stuurt berichten naar de editors. De editors zijn voorzien van functies die deze berichten kunnen afhandelen. De communicatie tussen de JFC/Swing Editor en de Meta-Environment vindt plaats via de ToolBus. Naast bovengenoemde basisfunctionaliteit moet de JFC/Swing Editor beschikken over functies die de aanroepen en berichten van Meta-Environment kunnen afhandelen. Het gaat hier om de functies voor onder andere:

- Het zetten van een focus (zie paragraaf 2.8);
- Het verwijderen van de focus;
- Openen van de ASF en SDF bestanden;
- Opslaan van de ASF en SDF bestanden;
- Een functie die de huidige activiteiten van de Meta-Environment op de statusbar van de JFC/Swing Editor toont (b.v. parsing, focus symbool, idle);
- Een functie voor het maken van menus. Vanuit deze menus kan men de parser aanroepen, de parse tree bekijken en de focus verplaatsen;
- Beëindigen van de JFC/Swing Editor sessie. Deze functie zorgt ervoor dat de JFC/Swing Editor wordt afgesloten;
- Plaatsen van de cursor;
- Een functie om de JFC/Swing Editor naar voren te brengen, zodat die zichtbaar wordt op het beeldscherm.

De JFC/Swing Editor moet ook in staat zijn om berichten te kunnen sturen naar de Meta-Environment. De Meta-Environment beschikt over functies die deze berichten kunnen afhandelen. De JFC/Swing Editor moet in staat zijn om de volgende berichten te kunnen sturen naar de Meta-Environment:

- Een bericht waarin de positie van de cursor staat vermeld. De Meta-Environment bepaalt aan de hand van de positie van de cursor waar de focus wordt gezet in de JFC/Swing Editor;
- De JFC/Swing Editor stuurt naar de Meta-Environment een bericht waarin staat vermeld dat er een wijziging heeft plaatsgevonden wanneer men een wijziging heeft aangebracht in de code;
- Een bericht waarin staat vermeld welke menu item men heeft gekozen. Vanuit de menus kunnen de gebruikers bijvoorbeeld de parser aanroepen, de parse tree bekijken, een term bewerken, de test uitvoeren en de focus verplaatsen.

Na de implementatie van bovengenoemde functionaliteit wordt de JFC/Swing Editor voorzien van syntax highlighting.

2.5 Onderzoeksvragen

Het onderzoek is gebaseerd op toegepaste research. Om de genoemde problemen gestructureerd op te lossen heb ik de volgende onderzoeksvragen gedefinieerd:

- Biedt JFC/Swing mogelijkheden voor het ontwikkelen van een editor die platformonafhankelijk is, met basisfunctionaliteit zoals Copy/Cut/Paste, Undo/Redo en Find/Replace?
- Op welke manier kan deze editor communiceren met de Meta-Environment?
- Hoe kan deze editor voorzien worden van syntax highlighting?

2.6 Verwachte resultaten

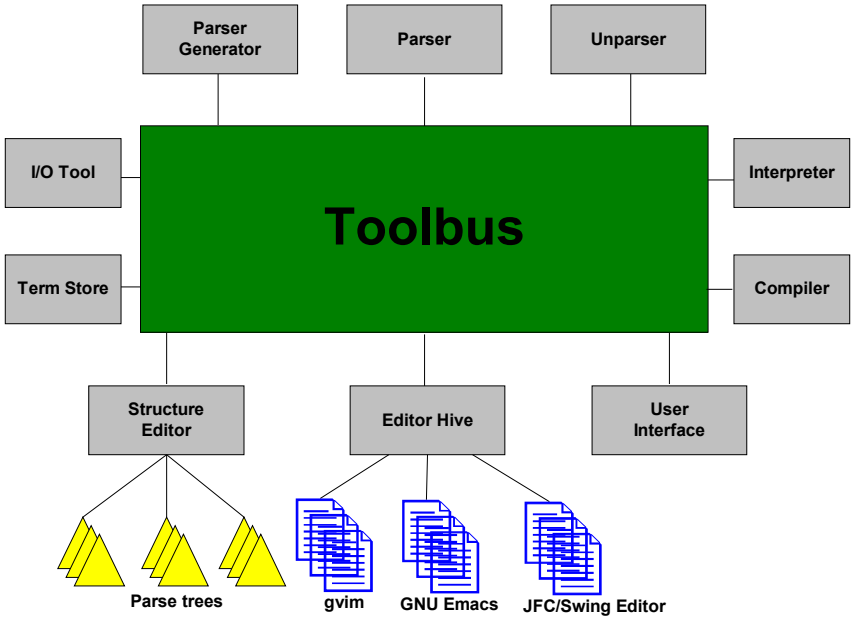
Als resultaat van het onderzoek wordt er een JFC/Swing Editor ontwikkeld die gekoppeld is aan de Meta-Environment. De JFC/Swing Editor moet dezelfde mogelijkheden bieden als de huidige externe editors (GNU Emacs en Vim) en over basisfunctionaliteit beschikken zoals Copy/Cut/Past, Undo/Redo en Find/Replace, dit om het gebruikersgemak van de eindgebruiker te verhogen.

Daarnaast moet de JFC/Swing Editor op alle platformen (zoals Microsoft Windows, Linux en Mac OSX) waarvoor de Java Virtual Machine beschikbaar is kunnen draaien zonder aanpassingen aan deze editor te brengen. Als laatste moet de JFC/Swing Editor voorzien worden van syntax highlighting.

2.7 Tools en technieken

Binnen het CWI worden veel tools en technieken gebruikt voor het koppelen van de componenten aan de Meta-Environment. ToolBus (zie paragraaf 2.2) wordt gebruikt voor de integratie van de JFC/Swing met Meta-Environment. In figuur 2 is te zien hoe de JFC/Swing Editor gekoppeld is aan de Meta-Environment. De Meta-Environment beschikt over de tool Structure Editor(SE). Voor elke editor die een goed geparseerde tekst bevat, is er een parse tree aanwezig in de SE(zie paragraaf 2.8). Naast ToolBus wordt ook gebruik gemaakt van Editor Hive (zie paragraaf 2.9) om de editors niet direct te koppelen aan de ToolBus.

Daarnaast wordt ApiGen [3] gebruikt voor genereren van C en Java API's en ADT voor het genereren van datastructuren (zie ook paragraaf 2.10)



Figuur 2: architectuur van ASF+SDF Meta-Environment

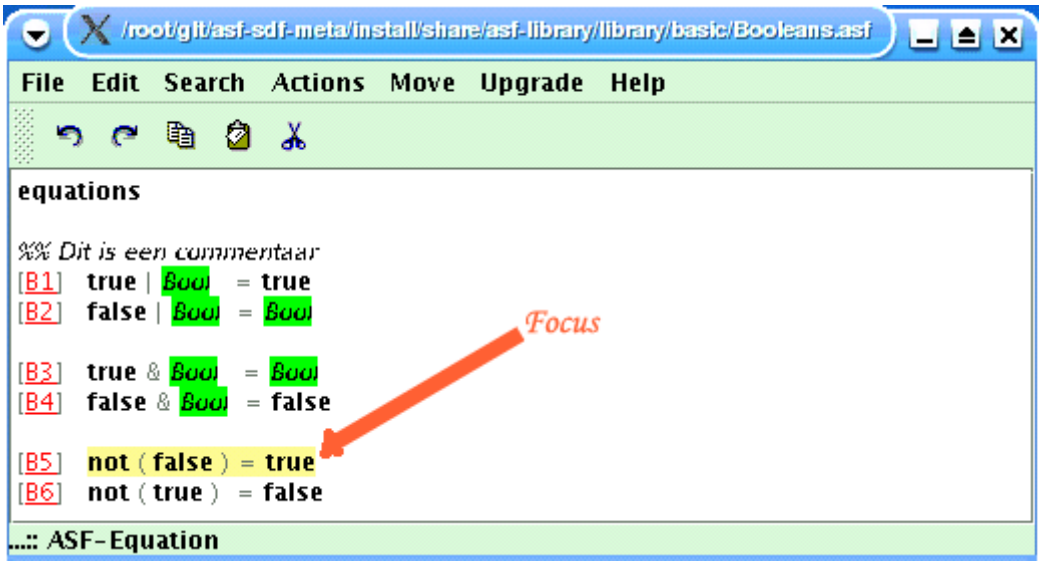
2.8 Structure Editor

De tekst editor en de Structure Editor(SE) wisselen informatie uit tijdens een bewerk sessie om bijvoorbeeld de focus en de sort te bepalen. De parse tree bevat informatie over de tekst die men aan het bewerken is:

- Welk type elke sub boom heeft;
- Waar elke sub boom begint en eindigt in termen van regels, kolommen, beginposities en lengtes van sub bomen (positie informatie).

Om de focus in de editor te kunnen zetten, wordt informatie gebruikt die de parse tree bevat. Men klikt met de muisknop op een woord in de editor. Een bericht waarin de positie van de cursor staat vermeld wordt gestuurd naar de Meta-Environment. De SE bepaalt aan de hand van de de positie van de cursor in welke sub boom het woord zich bevindt. De SE geeft vervolgens aan de editor de positie informatie door en welk type sub boom het is. De editor zet de focus door gebruik te maken van deze positie informatie en toont op de statusbalk welk type sub boom de focus is.

In figuur 3 is een voorbeeld te zien van een focus. Op de statusbalk in figuur 3 staat het type sub boom vermeld. De focus wordt gerealiseerd door de tekst te voorzien van een achtergrondkleur zoals dat het geval is in figuur 3.

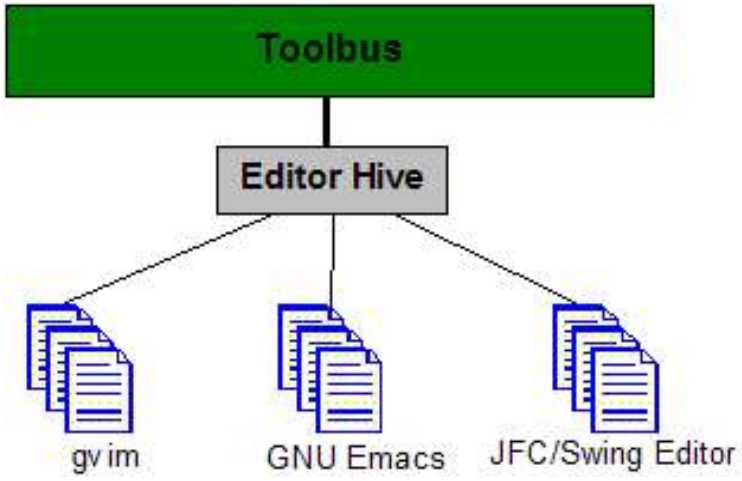


Figuur 3: een voorbeeld van focus in de JFC/Swing Editor

2.9 Editor Hive

De JFC/Swing Editor is niet direct gekoppeld aan de ToolBus. In het verleden had elke editor sessie een eigen ToolBus proces. Om elke bewerk sessie te kunnen afhandelen, moest veel administratie gedaan worden in de ToolBus. De ToolBus is bedoeld voor de coördinatie van processen en niet voor de berekeningen.

Tussen de ToolBus en de JFC/Swing Editor wordt de Editor Hive tool gebruikt. Dit tool is geïntroduceerd om de editors niet direct te koppelen aan de ToolBus. Hiermee wordt voorkomen dat de ToolBus berekeningen gaat uitvoeren voor elke editor sessie. Editor Hive zorgt voor de communicatie met ToolBus en administratie van editors. Nu is er maar één proces nodig om de sessies af te handelen (zie bijlage A). In figuur 4 is de Editor Hive tool en zijn eigen ToolBus proces weergegeven. Zoals te zien is in figuur 4 kunnen meer editors gekoppeld worden aan de Editor Hive.



Figuur 4: Editor-hive

2.10 ApiGen

Application Program Interface Generator (ApiGen) is een tool om C en Java API's te kunnen genereren voor de componenten die gekoppeld worden aan de Meta-Evironment. Een Annotated Data Type (ADT) wordt gebruikt voor het beschrijven van de datastructuren van de berichten die tussen de componenten van de Meta-Environment worden gestuurd. ApiGen gebruikt een ADT als input om Java en C API's te genereren. Met deze API's kunnen de waarden in de berichten worden gelezen. De API's worden ook gebruikt om berichten te creëren die de componenten van de Meta-Environment aan elkaar sturen. ApiGen is generieker. Men kan gebruik maken van ApiGen om bibliotheken te genereren waar men tegen kan programmeren.

In figuur 5 is een voorbeeld van ADT te zien. Deze ADT beschrijft de tekstopmaak voor de code die wordt bewerkt in de JFC/Swing Editor. In deze ADT staan vier tekst categorieën beschreven. Dat zijn de focus, selection, normal en extern. Voor deze tekstcategorieën zijn tekstattributen beschreven zoals foreground-color(voorgrondkleur), background-color (achtergrondkleur), size (lettergrootte), font (lettertype) en style. Onder style vallen bold (vet), italics (cursief) en underlined (onderstreept).

```
[
constructor(Property, text-category, text-category(<category(TextCategoryName)>,<attributes
(TextAttributes)>)),
constructor(TextCategoryName, focus, focus),
constructor(TextCategoryName, selection, selection),
constructor(TextCategoryName, normal, normal),
constructor(TextCategoryName, extern, extern(<name(str)>)),list(TextAttributes, TextAttribute),
constructor(TextAttribute, foreground-color, foreground-color(<color(Color)>)),
constructor(TextAttribute, background-color, background-color(<color(Color)>)),
constructor(TextAttribute, style, style(<style(TextStyle)>)),
constructor(TextAttribute, font, font(<name(str)>)),
constructor(TextAttribute, size, size(<points(int)>)),
constructor(TextStyle, bold, bold),
constructor(TextStyle, italics, italics),
constructor(TextStyle, underlined, underlined),
constructor(Color, rgb, rgb(<red(int)>,<green(int)>,<blue(int)>))
]
```

Figuur 5: ADT tekstopmaak

3 JFC/Swing Editor

3.1 Gebruikte technologieën

Voor de uitvoering van dit project is er gebruik gemaakt van veel technieken en tools die binnen het CWI zijn ontwikkeld. Voor het ontwerp en de realisatie van JFC/Swing Editor is gebruik gemaakt van JFC/Swing [8].

Voor de communicatie tussen de JFC/Swing Editor en de ASF+SDF Meta-Environment is ToolBus (zie paragraaf 2.2) gebruikt. Het ToolBus script is gebruikt voor het genereren van de Java klassen die voor de communicatie zorgen tussen de Meta-Environment en de JFC/Swing Editor. De tool Tifstojava is gebruikt om de volgende Java bestanden te genereren:

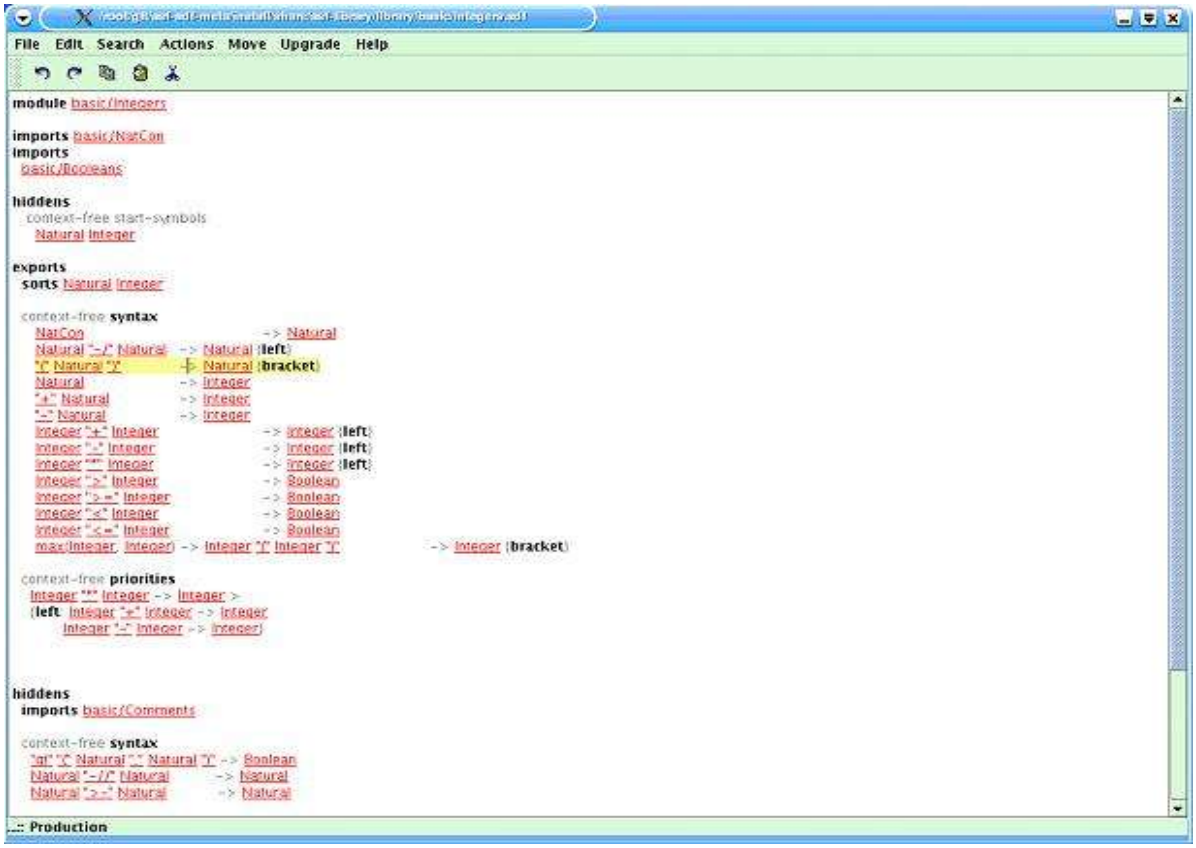
- SwingEditorTif.java is de Java interface die de functionaliteit van de JFC/Swing Editor beschrijft;
- SwingEditorTool.java is een abstracte implementatie van SwingEditorTif.java. SwingEditorTool.java bevat de code die de input/output signatuur van de JFC/Swing Editor controleert. Deze code handelt alle binnenkomende berichten af door elk bericht naar de juiste methode te sturen;
- SwingEditorBridge.java is een concrete klasse die van de klasse SwingEditorTool.java overerft. Deze klasse wordt als brug gebruikt tussen de JFC/Swing Editor en de Meta-Environment.

Verder is er gebruik gemaakt van het ADT formaat voor het beschrijven van de datastructuren van alle berichten die tussen de JFC/Swing Editor en de Meta-Environment worden gestuurd. ApiGen is gebruikt voor het genereren van Java API's. Met deze API's kunnen de waarden in de berichten gelezen worden. Deze API's worden ook gebruikt om berichten te kunnen creëren die de JFC/Swing Editor aan de Meta-Environment stuurt.

3.2 JFC/Swing Editor GUI

JFC/Swing bevat een groot aantal klassen die onderdelen van grafische schermen representeren, zoals panels, scrollbars, knoppen, editors en menu's. Dat zijn precies de onderdelen die nodig zijn voor het ontwikkelen van de JFC/Swing Editor (zie figuur 6).

De klassen van JFC/Swing die zijn gebruikt voor de realisatie van de JFC/Swing Editor zijn ImageIcon, JButton, JFrame, JLabel, JMenu, JMenuBar, JMenuItem, JPanel, JScrollPane, JSeparator, JTextPane en JToolBar.



Figuur 6: JFC/Swing Editor.

Het was niet nodig om zelf klassen te schrijven die de grafische schermen representeren. Dat brengt veel voordelen met zich mee aangezien dat veel tijd en kennis vereist. De GUI van de JFC/Swing Editor bestaat uit:

- Een menubar, menu's en onderliggende menu items (zie paragraaf 3.6);
- Een toolbar en onderliggende knoppen undo, redo, copy, paste en cut (zie figuur 7);
- Een tekstcomponent voor toevoegen en bewerken van tekst (zie paragraaf 4.3);
- Een statusbar die de huidige activiteiten van de Meta-Environment toont (b.v. parsing, focus symbool, idle);


Figuur 7: JFC/Swing Editor toolbar.

3.3 Copy/Cut/Paste

JFC/Swing beschikt over vijf tekstcomponenten, samen met klassen en interfaces die ondersteuning geven. De tekstcomponent die wordt gebruikt voor de JFC/Swing Editor is JTextPane. Deze tekstcomponent beschikt over de functionaliteit Copy/Cut/Paste. Het is dus niet nodig om hiervoor speciale functies te schijven. De clipboard functies JTextPane.cut(), JTextPane.copy() en JTextPane.paste() dienen wel aangeroepen te worden.

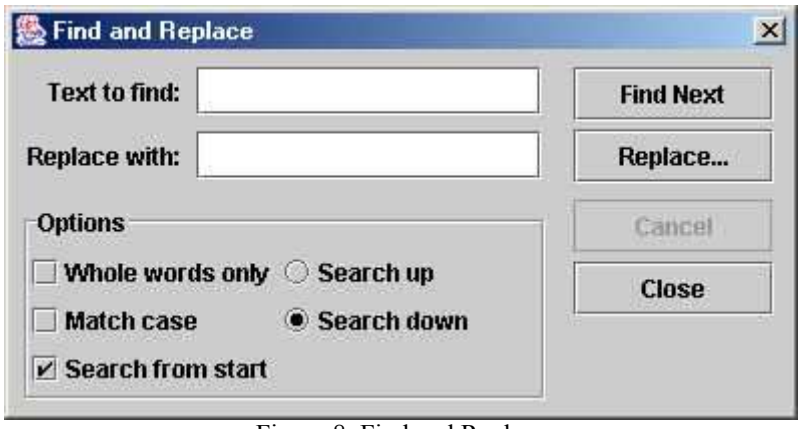
3.4 Undo/Redo

De tekstcomponent JTextPane ondersteunt naast de tekstverwerker operaties Copy/Cut/Paste ook oneindig de undo/redo operaties[9]. Om deze operaties te kunnen ondersteunen moet deze

tekstcomponent elke bewerking die plaats vindt en de volgorde daarvan onthouden. In de JFC/Swing Editor wordt gebruik gemaakt van de JFC/Swing UndoManager klasse om de lijsten van bewerkingen te kunnen beheren. UndoManager is ook gebruikt om aan de Meta-Environment door te geven of er een wijziging heeft plaatsgevonden in de Meta-Environment bestanden.

3.5 Find/Replace

De JFC/Swing Editor biedt functionaliteit voor het zoeken en vervangen van tekst (zie figuur 8). Om dit te kunnen ondersteunen, is er gebruik gemaakt van open source [6]. In de tekstvakken kan de gezochte tekst en de nieuwe tekst ingetypt worden.



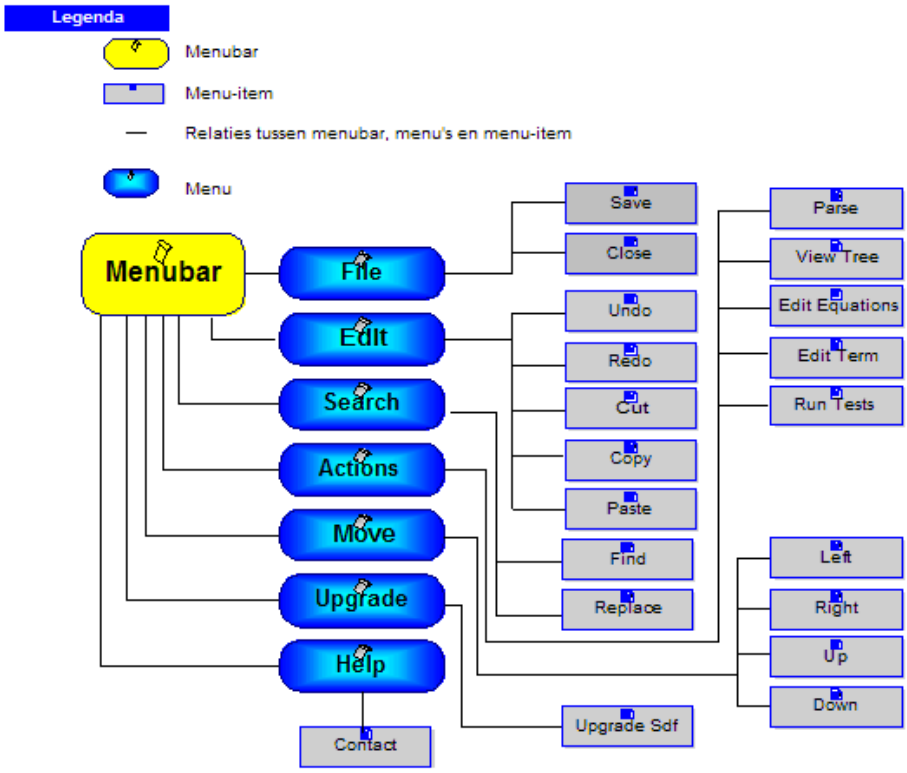
Figuur 8: Find and Replace

In het dialoogkader zijn een aantal extra opties:

Optie	Functie
Whole words only	Gezocht wordt naar gehele worden(eventueel tussen leestekens).
Match case	Gezocht wordt naar woorden met hoofd- en kleine letter, exact zoals ingetypt.
Search from start	Gezocht wordt vanaf het begin van de tekst.
Search up	Gezocht wordt van beneden naar boven.
Search down	Gezocht wordt van boven naar beneden.

3.6 Menustructuur van JFC/Swing Editor

In figuur 9 is de menustructuur van de JFC/Swing Editor weergegeven. De menu's File, Edit, Search en Help zijn statisch. De menu's Actions, Move, Upgrade en onderliggende menu items worden door de Meta-Environment aan de JFC/Swing Editor doorgegeven.



Figuur 9: Menubar, menu's en menu-items.

In figuur 10 is een voorbeeld weergegeven van een bericht dat door de Meta-Environment wordt doorgegeven aan de JFC/Swing Editor bij het openen van een SDF bestand. In dat bericht staan de menu's Actions, Move, Upgrade en de onderliggende menu items vermeld. De menu Actions beschikt over de menu items Parse, View Tree, Edit Equations, Edit Term, Run Tests. De menu Move beschikt over de menu items Left, Right, Up en Down. Deze menu items worden gebruikt om de focus te kunnen verplaatsen. De menu Upgrade beschikt over de menu item Upgrade Sdf.

Om deze menu's en menu items uit het bericht te kunnen lezen, wordt er gebruik gemaakt van de API's die gegenereerd zijn door ApiGen. Ook is er gebruik gemaakt van de klasse MenuBar. Deze klasse zorgt voor het creëren van een menubar volgens het bericht waarin de menustructuur staat vermeld.

```
[menu(["Actions","Parse"]),
menu(["Actions","View Tree"]),
menu(["Actions","Edit Equations"]),
menu(["Actions","Edit Term"]),
menu(["Actions","Run Tests"]),
menu(["Move","Left"]),
menu(["Move","Right"]),
menu(["Move","Up"]),
menu(["Move","Down"]),
menu(["Upgrade","Upgrade Sdf"])]
```

Figuur 10: Menu's Actions, Move en upgrade

3.7 Sneltoetsen

In JFC/Swing Editor kan men gebruik maken van een aantal toetsencombinaties om snel een bepaalde actie uit te voeren. De klasse JMenuItem van JFC/Swing beschikt over een functie die men kan

gebruiken voor sneltoetsen. Men kan gebruik maken van de volgende sneltoetsen van de JFC/Swing Editor:

Sneltoets	Actie
Ctrl+S	Het geopende bestand opslaan.
Ctrl+Z	Ongedaan maken van acties.
Ctrl+R	Heruitvoeren van ongedaan gemaakte acties.
Ctrl+X	Geselecteerde tekst verwijderen.
Ctrl+C	Geselecteerde tekst kopiëren.
Ctrl+V	Gekopieerde tekst plakken.
Ctrl+F	Tekst zoeken en vervangen.
Ctrl+H	Tekst zoeken.

3.8 Model View Controller

De JFC/Swing editor is ontwikkeld volgens het Model View Controller (MVC) [14] concept, waarbij een duidelijk onderscheid wordt gemaakt tussen de verschillende gebieden (zie figuur 11). Het MVC concept is ontwikkeld voor systemen die gebaseerd zijn op een GUI zoals het geval is met JFC/Swing Editor. De JFC/Swing Editor wordt opgedeeld in drie basis delen: het model, de view en de controller.

3.8.1 Het model

Het model bevat de interne logica en de datastructuur van de JFC/Swing Editor. Dit gedeelte is volledig onafhankelijk van de GUI van de JFC/Swing Editor. Binnen het model is gebruik gemaakt van de API's die gegenereerd zijn door ApiGen. Met deze API's worden de berichten die gestuurd worden van Meta-Environment naar de JFC/Swing Editor gelezen. Deze API's worden tevens gebruikt om berichten te kunnen genereren die JFC/Swing Editor stuurt naar de Meta-Environment.

3.8.2 De view

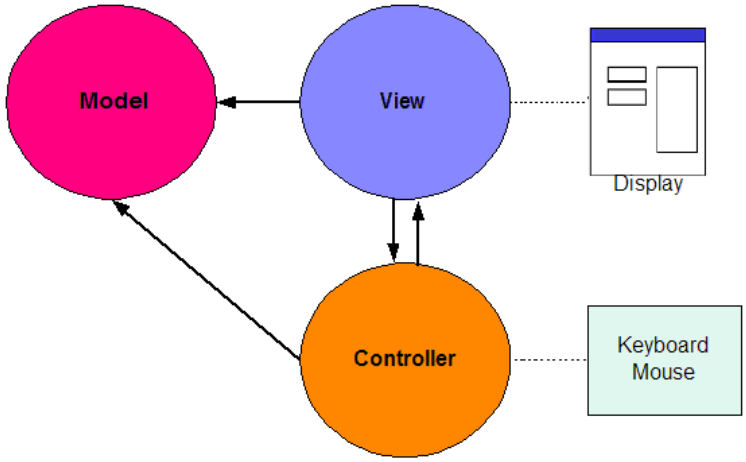
De view bevat de GUI onderdelen van de JFC/Swing Editor die de gebruiker ziet en kan manipuleren (zie figuur 6).

3.8.3 De controller

De controller is het onderdeel dat de gebruikersinteractie afvangt en vertaalt in opdrachten naar het model. De controller zorgt ook voor de interactie met de Meta-Environment. De controller implementeert de SwingEditorTif interface en bevat alle functies die worden aangeroepen door de Meta-Environment. Hieronder worden de functies genoemd die aangeroepen worden door de Meta-Environment:

- displayMessage: de Meta-Environment geeft aan de JFC/Swing Editor door welk bericht er getoond moet worden op de statusbalk. Op de statusbar worden dan de huidige activiteiten van de Meta-Environment getoond (b.v. parsing, focus symbool, idle);
- setFocus: de Meta-Environment geeft aan de JFC/Swing Editor door waar de focus gezet dient te worden;
- clearFocus: de Meta-Environment vraagt de JFC/Swing Editor de focus te verwijderen;
- editFile: de Meta-Environment geeft aan de JFC/Swing Editor de bestandsnaam door die geopend dient te worden;
- addActions: de Meta-Environment geeft aan de JFC/Swing Editor door welke menustructuur er gebruikt moet worden (zie paragraaf 3.6);
- killEditor: de Meta-Environment beëindigt de JFC/Swing Editor sessie door deze functie aan te roepen. De JFC/Swing Editor wordt afgesloten;

- writeContents: de Meta-Environment vraagt aan de JFC/Swing Editor om de inhoud naar het ASF of SDF bestand te schrijven. Op deze manier wordt het bestand opgeslagen;
- registerTextCategories: de Meta-Environment geeft aan de JFC/Swing Editor aan welke tekstopmaak er gebruikt moet worden voor focus, syntax highlighting en normale tekst;
- setCursorAtOffset: de Meta-Environment geeft aan de JFC/Swing Editor aan waar de cursor gezet dient te worden;
- highlightSlices: de Meta-Environment geeft aan de JFC/Swing Editor een lijst door van areas (gebieden) waar syntax highlighting toegepast dient te worden;
- editorToFront: de Meta-Environment brengt door deze functie aan te roepen de JFC/Swing Editor naar voren, zodat die zichtbaar op het beeldscherm wordt;
- rereadContents: de Meta-Environment vraagt de JFC/Swing Editor de inhoud van het bestand nog eens te lezen;



Figuur 11: MVC Model

De gebruiker van JFC/Swing Editor heeft alleen maar weet van de view van de JFC/Swing Editor. Wanneer er een interactie plaatsvindt tussen gebruiker en de JFC/Swing Editor gaat dat als volgt:

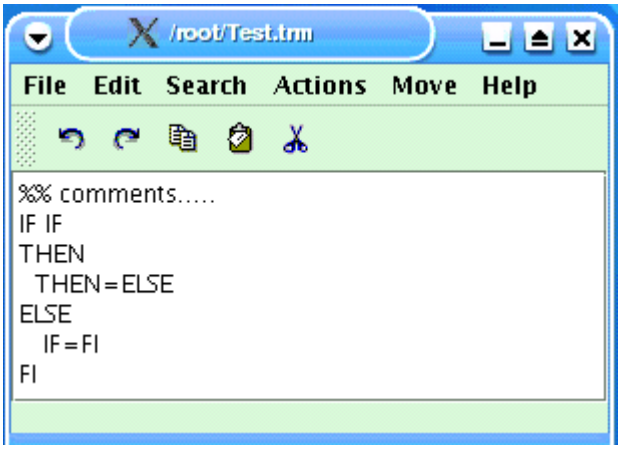
- De gebruiker manipuleert een onderdeel van de view;
- Deze manipulatie wordt afgevangen door de controller;
- De controller zorgt ervoor dat het bericht bij het juiste onderdeel van het model terechtkomt;
- De data in het model wordt veranderd door een interne berekening/manipulatie;
- Een bericht wordt gestuurd naar de relevante view zodat de nieuwe waarde aan de gebruiker getoond kan worden. Het bericht kan ook gestuurd worden naar de Meta-Environment.

Het verdelen van de JFC/Swing Editor in drie verschillende delen heeft een aantal voordelen. De belangrijkste worden hieronder opgenoemd:

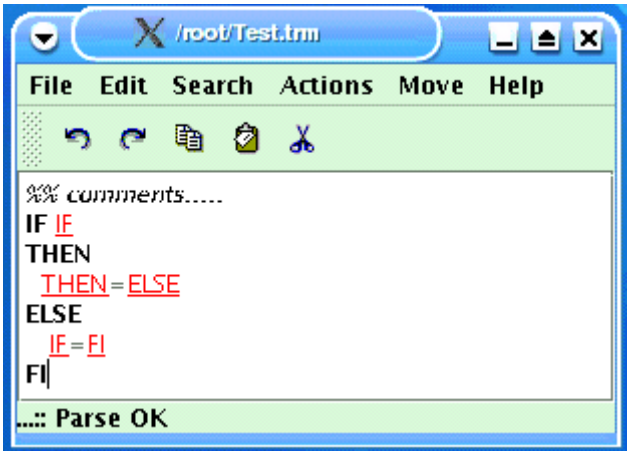
- Het is gemakkelijk om de GUI van de JFC/Swing Editor te veranderen vanwege de strikte scheiding tussen het model en de view;
- Ondersteunt het modulair ontwerpen van een applicatie vanwege de scheiding;
- Stimuleert hergebruik van onderdelen;
- Makkelijker debuggen vanwege het feit dat de fouten zich in het model voordoen. Deze fouten zijn vaak het gevolg van berekeningen, onverwachte combinaties van waarden of niet gewiste oude waarden. Omdat het model een losstaand geheel is, kan dit getest worden voordat het een onderdeel is van de hele applicatie. Eclipse [12] is gebruikt als ontwikkelomgeving voor de JFC/Swing Editor. Eclipse beschikt over een debugger. Deze debugger is ingezet tijdens ontwikkelen van de JFC/Swing Editor om de fouten in de code op te sporen.

4 Syntax highlighting

Syntax highlighting is een handige manier om de code verschillende eigenschappen mee te geven. Deze eigenschappen zijn bijvoorbeeld de achtergrondkleur en de voorgrondkleur van de tekst of het gebruikte font. Gebruikers kunnen de code in de ASF, SDF en term bestanden veel sneller lezen aangezien de syntax opgelicht wordt weergegeven. In figuur 12 is een voorbeeld te zien van code van programmeertaal PL1 zonder syntax highlighting en in figuur 13 is de code voorzien van syntax highlighting.



Figuur 12: een voorbeeld statement in PL1(zonder syntax highlighting)



Figuur 13: een voorbeeld statement in PL1(met syntax highlighting)

4.1 Tekstopmaak

Om syntax highlighting te kunnen realiseren, zijn er wijzigingen aangebracht in de Meta-Environment en het ToolBus script waarin de processen zijn beschreven (zie bijlage A). Bij het openen van een ASF, SDF of term bestand, geeft de Meta-Environment aan de JFC/Swing Editor door welke tekstopmaak voor zowel de normale tekst, syntax highlighting, selectie als focus gebruikt moet worden.

In de Meta-Environment is een tool die de configuration-manager heet. Deze tool leest configuratie bestanden in. Later kunnen andere tools die gekoppeld zijn aan de Meta-Environment vragen wat hun configuratie eigenschappen (properties) zijn. In de configuration-manager is de property 'text-category' toegevoegd. Er zijn drie vaste systeem categorieën(focus, selectie en normaal) en verder

een oneindig aantal andere (extern). Elke categorie beschikt over zijn eigen tekst attributen voor tekstopmaak. De Meta-Environment definieert de standaard tekstopmaak voor de systeem categorieën en voor een aantal algemene categorieën(Ambiguous, Lexicals, NonAlphanumericLiterals, AlphanumericLiterals en Comments) die in elke taal te vinden zijn.

In figuur 14 staat een bericht in de vorm van een datastructuur. De JFC/Swing Editor krijgt dit bericht van de Meta-Environment. In dit bericht staan de eigenschappen van tekstopmaak zoals achtergrondkleur, de kleur van de tekst(voorgrondkleur) en het gebruikte lettertype. Deze eigenschappen worden gebruikt voor syntax highlighting, focus, selectie en de normale tekst.

Als eerste wordt de code die bewerkt wordt in de JFC/Swing Editor voorzien van tekstopmaak (normale) zoals vermeld staat in het bericht voordat syntax highlighting wordt toegepast. Voor normale tekst lezen wij uit het bericht de volgende tekst eigenschappen:
- Tekstkleur: zwart(rgb(0,0,0));
- Achtergrondkleur: wit (rgb(255,255,255));
- lettertype: System;
- lettergrootte: 12.

Extern in het bericht bevat alle categorieën (Ambiguous, Lexicals, NonAlphanumericLiterals, AlphanumericLiterals en Comments) die gebruikt worden om syntax highlighting te kunnen realiseren. Om syntax van categorie Ambiguous te voorzien van syntax highlighting wordt de achtergrondkleur in rgb (red, green, blue) meegegeven. In dit geval is het rood. Comments staat voor het commentaar in de code. Om het commentaar te kunnen herkennen, wordt de tekst italics(cursief) weergegeven. Alle Lexicals in de code hebben de voorgrondkleur rood en zijn onderstreept. Verder is in het bericht te lezen dat de Variabelen in de code italics(cursief) en met achtergrondkleur groen(rgb (0, 255, 0)) worden weergegeven. Daarnaast worden de AlphanumericLiterals bold(vet) en de NonAlphanumericLiterals in rgb(97, 97, 97) weergegeven. Voor focus wordt de achtergrondkleur in rgb (238, 220, 130) weergegeven.

```
Categories:[
text-category(extern("Ambiguous"),[background-color(rgb(255,0,0))]),
text-category(extern("Comments"),[style(italics)]),
text-category(extern("Lexicals"),[style(underlined),foreground-color(rgb(255,0,0))]),
text-category(extern("Variables"),[style(italics),background-color(rgb(0,255,0))]),
text-category(extern("NonAlphanumericLiterals"),[foreground-color(rgb(97,97,97))]),
text-category(extern("AlphanumericLiterals"),[style(bold)]),
text-category(selection,[background-color(rgb(173,255,47))]),
text-category(focus,[background-color(rgb(238,220,130))]),
text-category(normal,[foreground-color(rgb(0,0,0)),background-color(rgb(255,255,255)),font
("System"),size(12)])]
```

Figuur 14: bericht tekstopmaak

4.2 Areas

Syntax highlighting wordt gerealiseerd door gebruik te maken van de positie informatie die de parse tree bevat. De JFC/Swing Editor krijgt een lijst waarin de positie informatie van de syntax die gehighlight moet worden.

Als de code geparseerd is, wordt er een keer over de boom in de SE gelopen om alle onderdelen van die boom op een vaste manier te categoriseren. Voor elke (programmeer)taal is die categorisatie hetzelfde. Het resultaat van de categorisatie is een lijst van 'slices', elke slice heeft een categorie naam en een lijst van posities(areas) die verwijzen naar de stukken tekst die in die categorie vallen. Het heet een slice omdat elke slice op een bepaalde manier een flinke hap uit de tekst is.

Zie figuur 15 voor een voorbeeld bericht waarin de areas per categorie staan vermeld. Er zijn 13 areas in het bericht vermeld van de categorie NonAlphanumericLiterals. In elke area zijn 6 waarden vermeld. Dat zijn onder andere:

- Beginlijn;
- Beginkolom;
- Eindlijn;
- Eindkolom;
- Beginpositie;
- Lengte.

De JFC/Swing Editor maakt alleen gebruik van de beginpositie en de lengte om de syntax te highlighten. De andere positie informatie wordt gebruikt door de editor GNU Emacs en Vim.

```
Slices: [
slice("NonAlphanumericLiterals",[
        area(12,13,12,26,165,13),
        area(12,0,12,12,152,12),
        area(9,37,9,38,141,1),
        area(9,36,9,37,140,1),
        area(9,28,9,29,132,1),
        area(9,23,9,24,127,1),
        area(9,12,9,14,116,2),
        area(8,36,8,37,102,1),
        area(8,35,8,36,101,1),
        area(8,28,8,29,94,1),
        area(8,23,8,24,89,1),
        area(8,12,8,14,78,2),
        area(6,0,6,12,45,12)]),
slice("Lexicals",[
        area(13,2,13,9,181,7),
        area(9,29,9,36,133,7),
        area(9,24,9,28,128,4),
        area(1,7,1,20,7,13)]),
slice("AlphanumericLiterals",[
        area(11,0,11,7,144,7),
        area(6,13,6,19,58,6),
        area(5,0,5,5,31,5),
        area(3,0,3,7,22,7),
        area(1,0,1,6,0,6)])]
```

Figuur 15: een voorbeeld van een aantal syntax highlighting areas

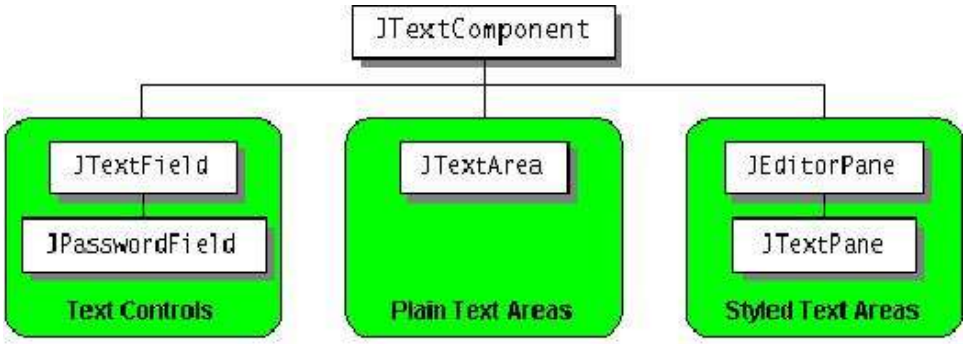
4.3 JFC/Swing tekstcomponenten

JFC/Swing beschikt over tekstcomponenten [9], samen met klassen en interfaces die ondersteuning geven (zie figuur 16). De tekstcomponenten zijn text field, password field, text area, editor pane en text pane.

Text field, password field en text area zijn “unstyled” tekstcomponenten. De tekstcomponenten text field en password field zijn niet geschikt om de bestanden van de Meta-Environment te kunnen bewerken omdat men in deze tekstcomponenten maar een regel tekst kan plaatsen. Het is niet mogelijk om delen van de tekst in een text area te voorzien van verschillende tekstopmaken. Bij syntax highlighting wordt gebruik gemaakt van verschillende tekstopmaken om delen van de code te kunnen highlighten.

Er zijn twee JFC/Swing klassen die tekst met opmaak ondersteunen: JEditorPane en zijn subklasse JTextPane. Deze twee alternatieven kunnen gebruikt worden voor de implementatie van syntax highlighting. JEditorPane is ontwikkeld voor verschillende soorten tekst en tekstopmaken(text/plain, text/html of text/rtf). JEditorPane wordt meestal gebruikt om HTML documenten en RTF (Rich Text Format) bestanden weer te geven.

Text pane kan behalve het weergeven van HTML en RTF bestanden ook gebruikt worden voor andere tekstformaten. Aan de text pane kunnen tekstopmaak attributen worden meegegeven en de posities waar deze tekstopmaak toegepast dienen te worden. Dat is precies wat nodig is om syntax highlighting te kunnen realiseren. De tekstopmaak wordt ondersteund door de JFC/Swing klassen SimpleAttributeSet en StyleConstants. StyleConstants krijgt een instantie van SimpleAttributeSet. In dit object worden dan alle tekstopmaak attributen gedefinieerd.



Figuur 16: JFC/Swing tekstcomponenten.

4.4 Tekstopmaak java klassen

Er zijn een aantal klassen geschreven om de tekstopmaak en de areas in figuur 14 en 15 te kunnen afhandelen (zie figuur 17). De klasse EditorStyle beschikt over een functie die alle opmaak attributen vastlegt en een functie die deze opmaak attributen in de JFC/Swing Editor toepast. Er is gebruik gemaakt van de API's die gegenereerd zijn door ApiGen om de waarden in het bericht in figuur 14 en 15 te kunnen lezen.

In figuur 18 is de functie die alle opmaak attributen vastlegt weergegeven. De functie in figuur 19 wordt gebruikt om de tekstopmaak toe te passen.

Figuur 17: EditorStyle klassen

```
public void makeStyle(Property property) {
   textAttributes = (TextAttributes) property.getAttributes();
   if (property.getCategory().isExtern()) {
       categoryName = property.getCategory().getName();
    }
    while (!textAttributes.isEmpty()) {
       textAttribute = (TextAttribute) textAttributes.getFirst();
       if (textAttribute.hasStyle()) {
          setStyle(textAttribute.getStyle());
       }
       if (textAttribute.isForegroundColor()) {
          setForegroundColor(getColor(textAttribute.getColor()));
       }
       if (textAttribute.isBackgroundColor()) {
          setBackgroundColor(getColor(textAttribute.getColor()));
       }
       if (textAttribute.isFont()) {
          setFontFamily(textAttribute.getName());
       }
       if (textAttribute.hasPoints()) {
          setFontSize(textAttribute.getPoints());
       }
       textAttributes = (TextAttributes) textAttributes.getNext();
   }
}
```

Figuur 18: functie makeStyle van klasse EditorStyle.

```
public void setStyle(StyledDocument document, int offset, int length, boolean replace) {
   document.setCharacterAttributes(offset, length, attributeSet, replace);
}
```

Figuur 19: functie setStyle van klasse EditorStyle.

Daarnaast zijn er drie subklassen van EditorStyle. De subklassen zijn Focuser, Highlighter en NormalStyle. Focuser beschikt over functies die voor het opzetten en weghalen van een focus zorgen. NormalStyle zorgt voor de (normale) tekstopmaak van de code. De code die in de JFC/Swing Editor wordt bewerkt, wordt als eerste voorzien van normale tekstopmaak. Als de code geparseerd is, wordt de code voorzien van syntax highlighting. De klasse Highlighter beschikt over een functie die de areas in het bericht in figuur 15 leest. Deze functie zorgt ook ervoor dat de tekstopmaken op de juiste posities in de JFC/Swing Editor worden toegepast. In figuur 20 is de code van deze functie weergegeven.

```
public void setHighlight(StyledDocument document){
    ATermList  copyClices=slices;
    Slice slice;
        while (!copyClices.isEmpty()){
            slice= locationFactory.SliceFromTerm(copyClices.getFirst());
            if (slice.getId().equals(categoryName)){
                areas=(AreaAreas) slice.getAreas();
                if (areas!=null){
                    while (!areas.isEmpty()) {
                        Area area =(Area) areas.getFirst();
                        if (area.hasOffset() && area.hasLength()) {
                            setStyle(document, area.getOffset(), area.getLength());
                        }
                        areas =(AreaAreas)areas.getNext();
                    }
                }
            }
            copyClices=copyClices.getNext();
        }
}
```

Figuur 20: functie setHighlighter van de klasse Highlighter

5 Gerelateerd werk

De ASF en SDF bestanden werden in de oude Meta-Environment[5] bewerkt door gebruik te maken van de Generic Structure Editor. GSE is een generieke tekst en structuur editor. De mogelijkheden van GSE wat betreft het bewerken van ASF en SDF bestanden zijn beperkt en zijn daardoor vervangen door Vim en GNU Emacs.

Naast Vim en GNU Emacs zijn er nog een aantal open source editors die voorzien zijn van geavanceerde mogelijkheden die het gebruikersgemak van de eindgebruiker verhogen. Men kan deze editors koppelen aan de Meta-Environment om ASF, SDF en term bestanden te kunnen bewerken. Voorbeelden van deze editors zijn Eclipse[12] en jEdit[13].

5.1 jEdit

jEdit is een tekst editor die onder de platformen MacOS X, OS/2, Unix/Linux, VMS en Windows draait. De jEdit website biedt meer dan 80 plugins aan die men kan downloaden en installeren door gebruik te maken van de plugin manager. De plugins maken van jEdit een volledige ontwikkelomgeving met mogelijkheden zoals gebruik maken van een compiler, debuggen en unit testen van applicaties.

Zoals de JFC/Swing Editor beschikt jEdit over oneindig Undo/Redo, Copy/Cut/Paste en Find/Replace. jEdit beschikt over meer geavanceerde mogelijkheden zoals: syntax highlighting voor meer dan 80 (programmeer)talen, het openen van meerdere bestanden waarbij men gemakkelijk kan overschakkelen door middel van tabbladden, zoeken en vervangen in meerdere bestanden, reguliere expressies, folding(vouwen), auto aanvulling etc.

Dankzij de bovengenoemde geavanceerde mogelijkheden is jEdit beter geschikt dan de JFC/Swing Editor.

5.2 Eclipse

Eclipse is een raam werk voor de integratie van ontwikkel tools voor diverse omgevingen en platformen. Met Eclipse kunnen ontwikkelaars de tools van verschillende leveranciers integreren voor het ontwikkelen van nieuwe toepassingen. Eclipse heeft een open structuur waardoor het mogelijk is de functionaliteit uit te breiden door middel van plugins. Eclipse zelf is een verzameling van een aantal plugins die geïmplementeerd zijn in Java. Eclipse heeft een ingebouwde tekst editor die gebruikt kan worden voor het bewerken van ASF, SDF en term bestanden. De integratie van Eclipse met de Meta-Environment is besproken in [4].

6 Conclusie en aanbevelingen

6.1 Conclusie

Mijn onderzoek heeft geresulteerd in een JFC/Swing Editor. Deze editor is gekoppeld aan de Meta-Environment. De JFC/Swing Editor is ook voorzien van de functionaliteit zoals Copy/Cut/Paste, Undo/Redo en Find/Replace.

Tevens zijn alle genoemde functies die de aanroepen en berichten van de Meta-Environment kunnen afhandelen geïmplementeerd. De JFC/Swing Editor is in staat om onder andere een focus te zetten en te verwijderen, een menustructuur te maken volgens het bericht dat wordt verzonden door de Meta-Environment, een ASF, SDF of term bestand op te slaan, de huidige activiteiten van de Meta-Environment te tonen, enz. De communicatie tussen de JFC/Swing Editor en de Meta-Environment vindt plaats dankzij het gebruik van de ToolBus.

De JFC/Swing Editor is bovendien voorzien van syntax highlighting door gebruik te maken van areas en tekstopmaak die worden meegegeven door de Meta-Environment. De JFC/Swing Editor zelf heeft geen weet van de syntax. De Meta-Environment bepaalt de tekstopmaak voor syntax highlighting en de areas waar de syntax in de code staat, en geeft dat door aan de JFC/Swing Editor.

Daarnaast kan de JFC/Swing Editor op elke computer draaien waarvoor de Java Virtual Machine (JVM) beschikbaar is.

6.2 Aanbevelingen

Als eerste moet de JFC/Swing Editor goed getest worden voordat men besluit het in productie te nemen. In de toekomst kan men de JFC/Swing Editor voorzien van nieuwe functionaliteit zoals de mogelijkheid om snel teksten door middel van drag & drop principe te verplaatsen. Een andere functionaliteit die men kan implementeren is Auto-completion(Auto-aanvulling). Auto-completion is een assistent die verschijnt terwijl men code tikt en aanbiedt om het huidige woord te vervolledigen. Ook kan de JFC/Swing Editor voorzien worden van de mogelijkheid om te kunnen zoeken naar tekst (en) door gebruik te maken van reguliere expressies.
Kortom genoeg ideeën die men nog in de toekomst kan implementeren om het gebruikersgemak van de eindgebruiker van de JFC/Swing Editor te verhogen.

7 Evaluatie

In begin van mijn afstudeerstage zijn er een aantal doelen vastgesteld. Het belangrijkste doel was het in de praktijk brengen van kennis, opgedaan gedurende het afgelopen studiejaar. Tijdens het studiejaar is er kennis opgedaan met de technieken en tools zoals ToolBus, ApiGen en ADT. Deze kennis is gebruikt om de JFC/Swing Editor te integreren met de Meta-Environment.

Tijdens dit project heb ik ervaring opgedaan met JFC/Swing en kennis gemaakt met het begrip 'design patterns'. Door dit project heb ik beter gevoel gekregen voor het ontwikkelen van systemen die gebaseerd zijn op een GUI. Mijn programmeerstijl is ook verbeterd ten opzichte van het begin van dit project. Ik heb de nadruk niet gelegd op een encyclopedische kennis van de JFC/Swing API, maar op een praktisch gebruik van deze API zoals event driven programming(gebeurtenisgestuurde programmatie), gebruik van luisteraars (observer/observable-patroon) en scheiding tussen model, view en controller.

7.1 Normering

Omschrijving	Motivering	Normering
Kwaliteit van het onderzoeksresultaat	Alle gedefinieerde onderzoeksvragen zijn tijdens dit project behandeld. Mijn onderzoek heeft geresulteerd in een JFC/Swing Editor die communiceert met de Meta-Environment via de ToolBus en hiermee is een brede basis gelegd waarop verder gewerkt kan worden. De implementatie van de JFC/Swing Editor en de koppeling daarvan met de Meta-Environment hebben de meeste tijd in beslag genomen.	8
Kwaliteit van de scriptie	In deze scriptie is er geprobeerd zorg te gedragen voor een heldere formulering, wat het document voor een breed publiek toegankelijk maakt. Dat is voor een deel niet gelukt. In de toekomst zal ik meer nadruk leggen op het plannen en structureren van een scriptie en de stijl en formulering ervan.	5,5
Moeilijkheidsgraad van de onderzoeksvraag	In dit project zijn er veel technieken en tools gebruikt. Om de JFC/Swing Editor te kunnen implementeren is veel kennis nodig van het programmeren van een grafische gebruikersinterface en gebruik maken van moderne paradigma's zoals gebeurtenisgestuurd programmeren en het MVC concept.	8
Relevantie van de vakken uit de Master Software Engineering voor het uitvoeren van dit project	De vakken software constructie en software evolution waren meeste relevant voor het uitvoeren van dit project. Voor de uitvoering van dit project is er gebruik gemaakt van veel technieken en tools zoals ToolBus, ApiGen en ADT. Deze tools zijn uitgebreid behandeld in het vak software evolution.	8

8 Bibliografie

8.1 Papers

[1] M.G.J. van den Brand, A. van Deursen, J. Heering, H.A. de Jong, M. de Jonge, T. Kuipers, P. Klint, L. Moonen, P.A. Olivier, J. Scheerder, J.J.Vinju, E. Visser, and J. Visser. *The ASF+SDF Meta-Environment: a Component-Based Language Development Environment.* In R. Wilhelm, editor, CC'01, volume 2027 of LNCS, pages 365-370. Springer-Verlag, 2001.

[2] J.A.Bergstra and P. Klint. *The discrete time ToolBus - a software coordination architecture*. Science of Computer Programming, 31(2-3):205-229, 1998.

[3] H.A. de Jong and P.A. Olivier. *Generation of abstract programming interfaces from syntax definitions*. Journal of Logic and Algebraic Programming(JLAP) 59 (2004) pages 35-61.

[4] M.G.J. van den Brand, H.A. de Jong, P. Klint and A.T. Kooiker. *A Language Development Environment for Eclipse*. Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, Pages 55-59, ACM Press, Anaheim, California, 2003.

[5] P. Klint. *A meta-environment for generating programming environments*, ACM Transactions on Software Engineering and Methodology, 2:176–201, 1993.

8.2 Websites

[6] Find and Replace(http://www.calcom.de/eng/dev/cctxt.htm)

[7] Implementing Undo and Redo(http://java.sun.com/docs/books/tutorial/uiswing/components/undo)

[8] Java Foundation Classes (http://java.sun.com/products/jfc/).

[9] Using Text Components(http://java.sun.com/docs/books/tutorial/uiswing/components/text.html)

[10]Vim (http://www.vim.org).

[11]GNU Emacs (http://www.emacs.org).

[12]Eclipse (http://www.eclipse.org/).

[13]jEdit(http://www.jedit.org)

8.3 Boeken

[14]James W.Cooper, *Java Design Patterns, A Tutotial*, Addison Wesley, 2000, ISBN 0-201-48539-7

9 Bijlage A: JFC/Swing Editor Idef

De editor hive idef handelt alle berichten van en naar de Meta-Environment editor hive. Aan dit idef bestand zijn wijzigingen gebracht om syntax highlighting te kunnen ondersteunen. “Categories” representeert de tekstopmaak die de editor moet gebruiken en “Slices” bevat de areas (plaatsen) van de editor waar de tekstopmaak toegepast moet worden. “Categories” bevat ook de tekstopmaak van de normale tekst en die van de focus.

```
tool editor-hive is {
  command = "editor-hive"
}
process EditorHive is
let
  ActionList : list,
  Area : term,
  EditorId : term,
  Editor : str,
  Filename : str,
  Focus : term,
  Hive : editor-hive,
  Offset : int,
  MenuEvent : term,
  Message : str,
  Modified : term,
  SortName: str,
  Pid: int,
  Categories: term,
  Slices: term
in
  execute(editor-hive, Hive?)
  .
  (
    rec-msg(te-edit-text(EditorId?, Editor?, Filename?))
    . snd-do(Hive, edit-file(EditorId, Editor, Filename))
  +
    rec-msg(te-add-actions(EditorId?, ActionList?))
    . snd-do(Hive, add-actions(EditorId, ActionList))
  +
    rec-msg(te-write-contents(EditorId?))
    . snd-do(Hive, write-contents(EditorId))
  +
    rec-msg(te-reread-contents(EditorId?))
    . snd-do(Hive, reread-contents(EditorId))
  +
    rec-msg(te-is-modified(EditorId?))
    . snd-do(Hive, is-modified(EditorId))
  +
    rec-msg(te-set-focus(EditorId?, Area?))
    . snd-do(Hive, set-focus(EditorId, Area))
  +
    rec-msg(te-set-cursor-at-offset(EditorId?, Offset?))
    . snd-do(Hive, set-cursor-at-offset(EditorId, Offset))
  +
    rec-msg(te-clear-focus(EditorId))
```

```
  . snd-do(Hive, clear-focus(EditorId))
 +
  rec-msg(te-register-text-categories(EditorId, Categories?))
  . snd-do(Hive, register-text-categories(EditorId, Categories))
 +
  rec-msg(te-highlight-slices(EditorId, Slices?))
  . snd-do(Hive, highlight-slices(EditorId, Slices))
 +
  rec-msg(te-display-message(EditorId?, Message?))
  . snd-do(Hive, display-message(EditorId, Message))
 +
  rec-msg(te-editor-to-front(EditorId?))
  . snd-do(Hive, editor-to-front(EditorId))
 +
  rec-msg(te-kill-text-editor(EditorId?))
  . snd-do(Hive, kill-editor(EditorId))
 +
  rec-event(Hive, is-modified(EditorId?, Modified?))
  . snd-msg(te-is-modified(EditorId, Modified))
  . snd-ack-event(Hive, is-modified(EditorId, Modified))
 +
  rec-event(Hive, contents-written(EditorId?))
  . snd-msg(te-contents-written(EditorId))
  . snd-ack-event(Hive, contents-written(EditorId))
 +
  rec-event(Hive, contents-changed(EditorId?))
  . create(ContentsChanged(EditorId), Pid?)
  . snd-ack-event(Hive, contents-changed(EditorId))
 +
  rec-event(Hive, mouse-event(EditorId?, Offset?))
  . create(MouseEvent(EditorId, Offset), Pid?)
  . snd-ack-event(Hive, mouse-event(EditorId, Offset))
 +
  rec-event(Hive, menu-event(EditorId?, MenuEvent?))
  . create(MenuEvent(EditorId, MenuEvent), Pid?)
  . snd-ack-event(Hive, menu-event(EditorId, MenuEvent))
 +
  rec-event(Hive, editor-disconnected(EditorId?))
  . snd-msg(te-text-editor-disconnected(EditorId))
  . snd-ack-event(Hive, editor-disconnected(EditorId))
 )
 * delta
endlet
```