Projektdokumentation

Thema :
Programmierung einer Eingabemaske für VMEbus - Befehle

Erstellt Juni 2002

Ausbildungsbetrieb:

XXXXXXXXXXXXX

Auszubildender:

Ralph Eggert-Vockerodt
Thesdorfer Weg 186
25421 Pinneberg

Inhaltsverzeichnis

1. Begriffserklärung

VMEbus (**V**-ersatile **M**-odule **E**-urope **bus**) ist eine auf dem Industriestandard basierende Hardware, die von ***Motorola*** entwickelt wurde. Sie hat die Aufgabe, verschiedenste Spezialkarten (CPU-Karten, I/O-Karten, Controller-Karten, ...), die alle auf diesem VMEbus-Standard basieren, zu verbinden.
Das ermöglicht dem Anwender, sich leistungsfähige Echtzeit-Computer-Systeme zur Steuerung komplexer technischer Applikationen zusammenzustellen. Der VMEbus-Standard baut sehr stark auf Motorola Mikroprozessoren und -controllern auf.

DESY (**D**eutsches **E**lektronen **S**ynchrotron) ist eines der weltweit führenden Zentren für die Forschung an Teilchenbeschleunigern. Hier erforschen Physiker experimentell die Welt der Elementarteilchen. Mehrere hundert Physikerinnen und Physiker arbeiten heute in internationalen Experimentiergruppen zusammen. Sie entwickeln, bauen und betreiben in langjähriger gemeinsamer Arbeit haushohe komplexe Messgeräte und werten Millionen von Daten aus. Bei DESY arbeiten derzeit vier solcher Gruppen an dem großen unterirdischen Beschleunigerring HERA.

2. Einleitung

Im Forschungsinstitut DESY wird das VMEbus - System als Datenerfassungsystem an verschiedenen Forschungsprojekten eingesetzt.
Um die VMEbus-Systeme auf fehlerfreie Funktion zu testen, ist es bisher nötig VMEbus - Befehlssequenzen in einer komplexen Programmiersprache wie (z.B. C oder Assembler) zu entwickeln.
Die noch in der Entwicklung befindliche Applikation VMEbus - Sequencer soll diese Aufgabe für die Mitarbeiter in Zukunft erleichtern.
Ziel dieser Projektarbeit ist es eine grafischen Benutzeroberfläche für die Eingabe von Parametern von **VMEbus - Befehlen** zu entwickeln und in das Projekt VMEbus - Sequencer zu integrieren.

3. Ziele

3.1. Sachziel des Gesamtprojektes VMEbus - Squencer

Das **Sachziel** des DESY-internen Projektes **VMEbus - Squencer** ist eine Applikation die mithilfe einer einfach zu bedienenden grafischen Benutzeroberfläche die Schnittstellen zwischen Benutzer und VMEbus - Rechner bereitstellt.
Über diese Schnittstelle soll es den Mitarbeitern ermöglicht werden, Input/Output Karten für den VMEbus auf fehlerfreie Funktion zu testen, ohne jeweils für einen solchen Test den Programmcode in einer komplexen Programmiersprache (z.B. Assembler oder C) entwickeln zu müssen.
Ein solcher Test soll zukünftig mit Befehlssequenzen arbeiten, die der Benutzer in der einfach zu bedienenden grafischen Benutzeroberfläche erstellt hat.
Beim Start eines solchen Tests überträgt die Applikation diese Befehlssequenzen an den **VMEbus - Rechner**, der wiederum diese an die angesprochene **VMEbus - Karte** zur Verarbeitung weiterleitet.
In dieser Applikation werden diese Befehlssequenzen grafisch dargestellt, editiert, verwaltet und die Ergebnisse der Verarbeitung angezeigt.
Eine Analyse des betriebswirtschaftlichen Nutzens des Projektes ist leider nicht möglich, da dieses Projekt in einem Forschungsinstitut durchgeführt wird, wo nicht zu ermitteln ist, wer der Mitarbeiter die Applikation wie oft nutzen wird.

3.2. Sachziel des Teilprojektes Eingabemaske für VME - Befehle

Das **Sachziel** dieses Teilprojektes ist es, mit der Programmiersprache Java eine **Eingabemaske zu programmieren**, die dem Benutzer die Bearbeitung **einzelner VMEbus - Befehle**[1] ermöglicht. Hierfür werden grafische Elemente aus dem **Swing - Paket**[2] verwendet werden. Für die Bearbeitung solcher VMEbus - Befehle sollen für die erforderlichen Datentypen der VMEbus - Befehle geeignete, grafische Steuerelemente in dieser Eingabemaske eingesetzt werden. Da VMEbus - Befehle eine **variable Anzahl** von Parametern haben, ist es erforderlich, daß sich die angezeigten grafischen Steuerelemente in ihrer Anzahl und Beschaffenheit an den jeweiligen VMEbus - Befehl **anpassen.**

4. Ausgangslage

4.1. Anforderungen an das Gesamtprojekt vom Auftraggeber (DESY)

1. Die Reihenfolge der VMEbus - Befehle soll in einer Baumdarstellung dargestellt und verändert werden können. In dieser Baumdarstellung soll jeweils ein Knoten (siehe Abb. 1 Parent - hier noch als Platzhalter angezeigt) einen VMEbus - Befehl darstellen. Die Parameterwerte dieses VMEbus - Befehls können auch teilweise von den Unterknoten (siehe Abb. 2 Child1+2 - hier noch als Platzhalter angezeigt) angezeigt werden.
2. Die Parameter der einzelnen VMEbus - Befehle sollen editiert werden können.
3. Die Ergebnisse der Verarbeitung der VMEbus - Befehlssequenzen sollen angezeigt werden können.

4.2. Vorüberlegungen zur Integration des Teilprojektes in das Gesamtprojekt

Um das Teilprojekt sachgemäß in das Gesamtprojekt zu integrieren ist eine Analyse des Aufbaus des Gesamtprojektes hinsichtlich des Ein - und Ausgabekonzeptes und der bereits vorhandenen Klassenstruktur notwendig.

4.2.1.Das Ein - und Ausgabekonzept des Gesamtprojektes

Um dem Benutzer das erlernen der Bedienung zu erleichtern und ein effektives Arbeiten zu ermöglichen, wurden die verschiedenen einzelnen Eingabeelemente nach ihren Aufgaben in sogenannten Panels[3] zusammengefasst.
Die durch diese Panels geschaffene modulare Aufteilung bietet nicht nur dem Benutzer mehr Übersichtlichkeit, sondern es wird auch der gegebenenfalls anfallende Wartungsaufwand bei Änderungen dieser Software reduziert.

4.2.2. Auflistung der Panels mit ihren Aufgaben

TreePanel — **Aufgabe:** Darstellung der **VMEbus - Befehlssequenz**

Ein Eingabeelement, in dem alle Eingabeelemente gruppiert sind, die für die Darstellung und Editierung der **VMEbus - Befehlssequenzen** notwendig sind. Dieses Panel ist bereits im Gesamtprojekt vorhanden.

VMEEditPanel — **Aufgabe:** Darstellung und Möglichkeit zur Bearbeitung der Parameter eines einzelnen VMEbus - Befehls.

---

[1] siehe Anhang Tabelle1 für weitere Informationen
[2] Swing ist Teil der Java Foundation Classes. Mit Hilfe des Swing Paketes wird die Programmierung grafischer Benutzeroberflächen wie sie von Microsoft, Linux usw. bekannt sind wesentlich vereinfacht.
[3] Panel (deutsch: Schalttafel)

Dieses Panel wurde in dieser Projektarbeit entwickelt.
Ein Eingabeelement, in dem alle Eingabeelemente gruppiert sind, die notwendig sind, um die einzelnen Parameter eines VMEbus - Befehls zu editieren und die Änderungen abzuspeichern. Hier können die VMEbus - Befehlstypen ausgewählt und die für den jeweiligen Befehlstyp nötigen Parameterwerte eingegeben und editiert werden. Auch sind in diesem Panel die Bedienelemente vorhanden, um Änderungen der editierten Parameterwerte abzuspeichern.

VMEResultPanel **Aufgabe:** Darstellung der Ergebnisse der Verarbeitung.

Einem Ausgabeelement, in dem alle Ausgabeelemente gruppiert sind, die für eine Ausgabe der Ergebnisse der Verarbeitung nötig sind. Dieses Panel ist noch nicht im Gesamtprojekt vorhanden und soll später noch entwickelt werden.

5. Ist - Zustand

5.1. Entwicklungsstadium des Gesamtprojektes

Das Gesamtprojekt VMEbus - Sequencer hat eine bereits implementierte GUI[3], in der eine Befehlssequenz im Treepanel als Knoten (siehe Abb.1 Parent1 - Parent4) in der Baumdarstellung dargestellt wird. In dieser Baumdarstellung können bereits Knoten ausgewählt und damit für die Verarbeitung im VMEEditPanel selektiert werden.

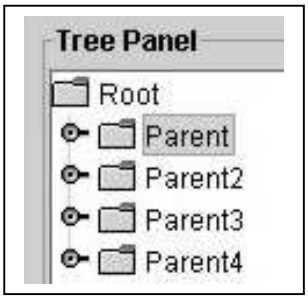


Abbildung 1
Selektierter Knoten "Parent"

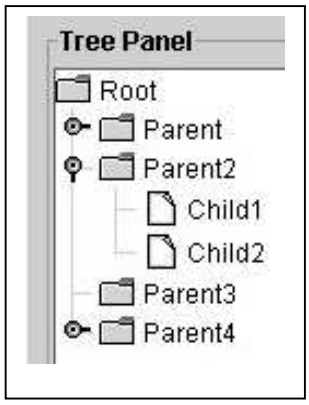


Abbildung 2
Aufgeklappter Knoten "Parent"

5.2. Daten vom Auftraggeber (DESY)

Der Auftraggeber hat dem Auftragnehmer eine Liste übergeben, (siehe Anhang Liste 1) die neben den VMEbus - Befehlsbezeichnungen und den dafür erforderlichen Datentypen auch die erforderlichen Eingabeelementtypen enthält.

[3] Graphical User Interface (deutsch: grafische Benutzeroberfläche)

6. Konkretisierung des Auftrages

Definition der Abgrenzung des Teilprojektes zum Gesamtprojekt
Diese Projektarbeit begrenzt sich auf die Entwicklung einer Eingabemaske für den Benutzer, um ihm die Bearbeitung der Parameterwerte eines im Treepanel ausgwählten VMEbus - Befehls zu ermöglichen.
Diese Eingabemaske soll soweit in das Gesamtprojekt integriert werden, bis die Änderung der Parameterwerte eines VME - Befehls im Arbeitsspeicher gespeichert wird.
Da jeder VMEbus - Befehlstyp eine andere Kombination von Parametern (siehe Anhang Tabelle 2) hat, ist es erforderlich dass sich die Eingabemaske an den vom Benutzer jeweilig ausgewählten VMEbus - Befehlstyp anpasst. Dies soll geschehen, indem einzelne GUI - Elemente für die Parametererfassung ein oder ausgeblendet werden, um Fehleingaben durch den Benutzer zu auszuschließen.

7. Lösungsansatz

Da das Gesamtprojekt in modular aufgebaut ist, wird auch ebenfalls das Teilprojekt, nämlich das VMEEditPanel, modular aufgebaut. Hier erscheint eine Trennung zwischen Eingabeelementen für die Parameterwerte und den Bedienelementen zum Abspeichern dieser Parameterwerte angebracht.
Deshalb werden die Eingabeelemente für die Parameterwerte in einem eigenen Panel (VMECommandControlPanel siehe Abb. 2) zusammengefasst und die Bedienelemente ebenfalls in einem eigenen Panel (VMEButtonPanel siehe Abb. 2).
Um diese beiden Panels auf dem Benutzeroberfläche platzieren zu können, ist ebenfalls wieder ein Panel (VMEEditPanel siehe Abb. 2) nötig. Das VMEEditPanel (siehe Abb. 2) hat nur die Aufgabe, die beiden anderen Panels zu platzieren.

Schematische Darstellung der benötigten Panels (siehe Abb. 2):

1. Das VMEEditPanel, welches die beiden anderen Panels platziert.

2. Das VMECommandControlPanel, in dem die Eingabeelemente zur Parameterwerterfassung zusammengefasst sind.

3. Das VMEButtonPanel in dem die Eingabeelemente zum Abspeichern und zu Zurücksetzen zusammengefasst sind.

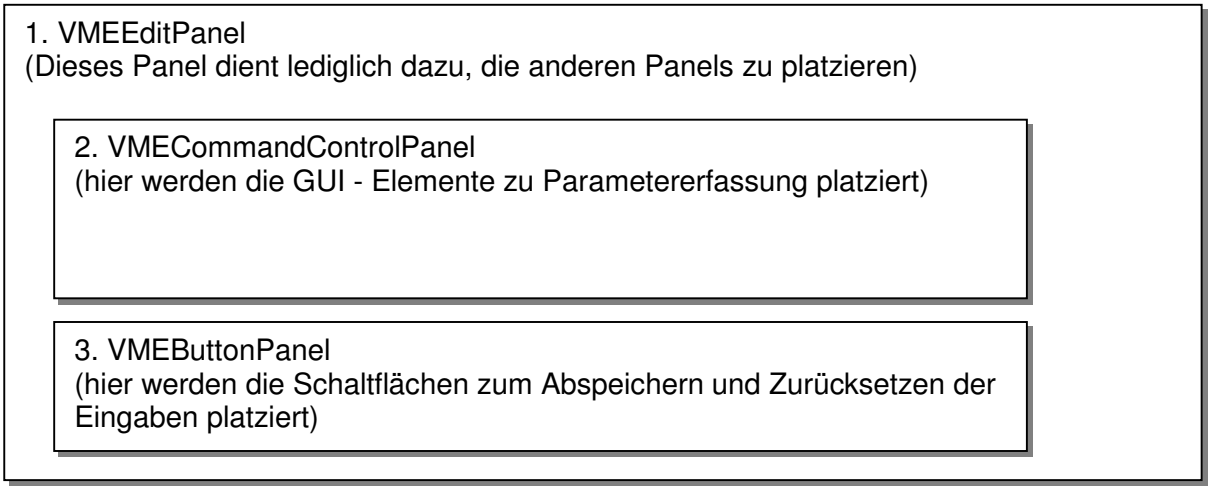


Abbildung 2 - Grafische Darstellung der Anordnung der Panels

8. Erarbeitung einer detaillierten Spezifikation

8.1. Analyse der vom Auftraggeber erhaltenen Liste

Ist diese Liste für die spätere Entwicklungsarbeit geeignet ?

Diese Liste enthält lediglich eine Aufzählung mit Parameterbezeichnung, erforderlichen Datentypen und Eingabeelementen für die Parameter.
Sie ist wenig übersichtlich und daher für die spätere Entwicklungsarbeit nicht geeignet. Sie wird in Entscheidungstabellen (siehe Anhang Tabelle1 und Tabelle2) umgesetzt, welche die Zuordnung der Parameterwerte und ihren Datentypen zu den Befehlstypen übersichtlicher und für die spätere Softwareentwicklung geeignet darstellt.
Diese Entscheidungstabellen werden Arbeitsgrundlage für das zu entwickelnde VMECommandControlPanel, (siehe Abb. 2) in der die Eingabeelemente für die Parameterwerte zusammengefasst werden sollen.

8.2. Ermittlung der GUI - Elemente zur Steuerung des VMEEditPanels

Wie können nun die geänderten Parameterwerte vom Benutzer abgespeichert werden ?

Damit der Benutzer das Panel in der späteren Anwendung sachgerecht nutzen kann, sind nicht nur Eingabeelemente für Parameterwerte notwendig, sondern auch Eingabeelemente zum Abspeichern und gegebenenfalls zum Zurücksetzen der Eingaben. Für diese Aufgaben eignen sich Schaltflächen. Diese Schaltflächen sollen aber nicht direkt mit den anderen Eingabeelementen zur Parametererfassung in einem Panel zusammengefasst werden, um bei einer möglichen Änderung der Benutzeroberfläche der Applikation den Wartungsaufwand durch eine modulare Programmierweise gering zu halten. Daher werden die Schaltflächen ebenfalls zu einem eigenen Panel, dem VMEButtonPanel (siehe Abb. 2) zusammengefasst. Diese Schaltflächen sollen auch dazu dienen, die gegenwärtige Gültigkeit der Eingaben wiederzuspiegeln, indem die Schaltfläche zum speichern bei ungültigen Eingaben als nicht bedienbar dargestellt wird.
Um dieses Problem zu beschreiben, wird ebenfalls eine Entscheidungstabelle (siehe Anhang Tabelle 3) erstellt.

8.3. Erstellen einer Skizze des zu erstellenden Panels für die Parametererfassung

Als Arbeitsgrundlage für diese Skizze dient die Entscheidungstabelle (Tabelle 1 siehe Anhang), in der die erforderlichen Parameter für VMEbus - Befehle aufgeführt sind. Zunächst werden alle in der Tabelle 1 aufgeführten Eingabeelemente für die Parameterwerte in die Skizze (siehe Anhang Skizze 2) übertragen.

Vorüberlegung:
Welche grafischen Elemente sind nötig, um dem Benutzer ein sachgemäßes Arbeiten zu ermöglichen ?

Die Eingabeelemente für die Parameterwerte und Bedienelemente allein reichen nicht aus, um mit diesem Panel sachgemäß arbeiten zu können.
Die Eingabeelemente benötigen Bezeichnungen, damit der Benutzer erkennen kann, um welchen VMEbus - Parameterwert es sich bei diesem Eingabeelement handelt.

Um eine Skizze erstellen zu können, die auch als Arbeitsgrundlage für die Programmierung dienen wird, muss hier entschieden werden, wie das Layout der grafischen Elemente erfolgen soll.

Daher ist hier folgende Frage zu beantworten:

Wie soll das Layout realisiert werden ?

Um grafische Eingabe- und Bedienelemente darstellen und positionieren zu können, werden in Java sogenannte Layoutmanager verwendet. Hier ist eine Entscheidung für den Gridbaglayoutmanager gefallen, weil dieser als einziger in der Lage ist, Elemente verschiedener Größe flexibel zu platzieren.
Dieser Gridbaglayoutmanager arbeitet mit einem Raster, in dem er die Elemente platziert. Dieses Raster wird mithilfe der noch zu erstellenden Skizze für das VMEEditPanel ermittelt. Da diese Skizze als Arbeitsgrundlage für das spätere Programmieren des Layoutmanagements werden soll, ist es sinnvoll, gleich an dieser Stelle neben dem Raster für den Layoutmanager gleich auch die erforderlichen Parameterwerte zu ermitteln und auch direkt in der Skizze festzuhalten.

Erarbeitung der Skizze des zu realisierenden Layouts

Aufgrund der Entscheidungstabellen (Tabelle 1 siehe Anhang), wird nun die Anordnung der Eingabeelemente und die dafür erforderlichen Bezeichnungselemente erarbeitet. Danach wird das für diese Anordnung erforderliche Raster ermittelt. Mithilfe dieses Rasters werden nun die Parameterwerte für den Layoutmanager ermittelt und auf der Skizze 2 (siehe Anhang) festgehalten.

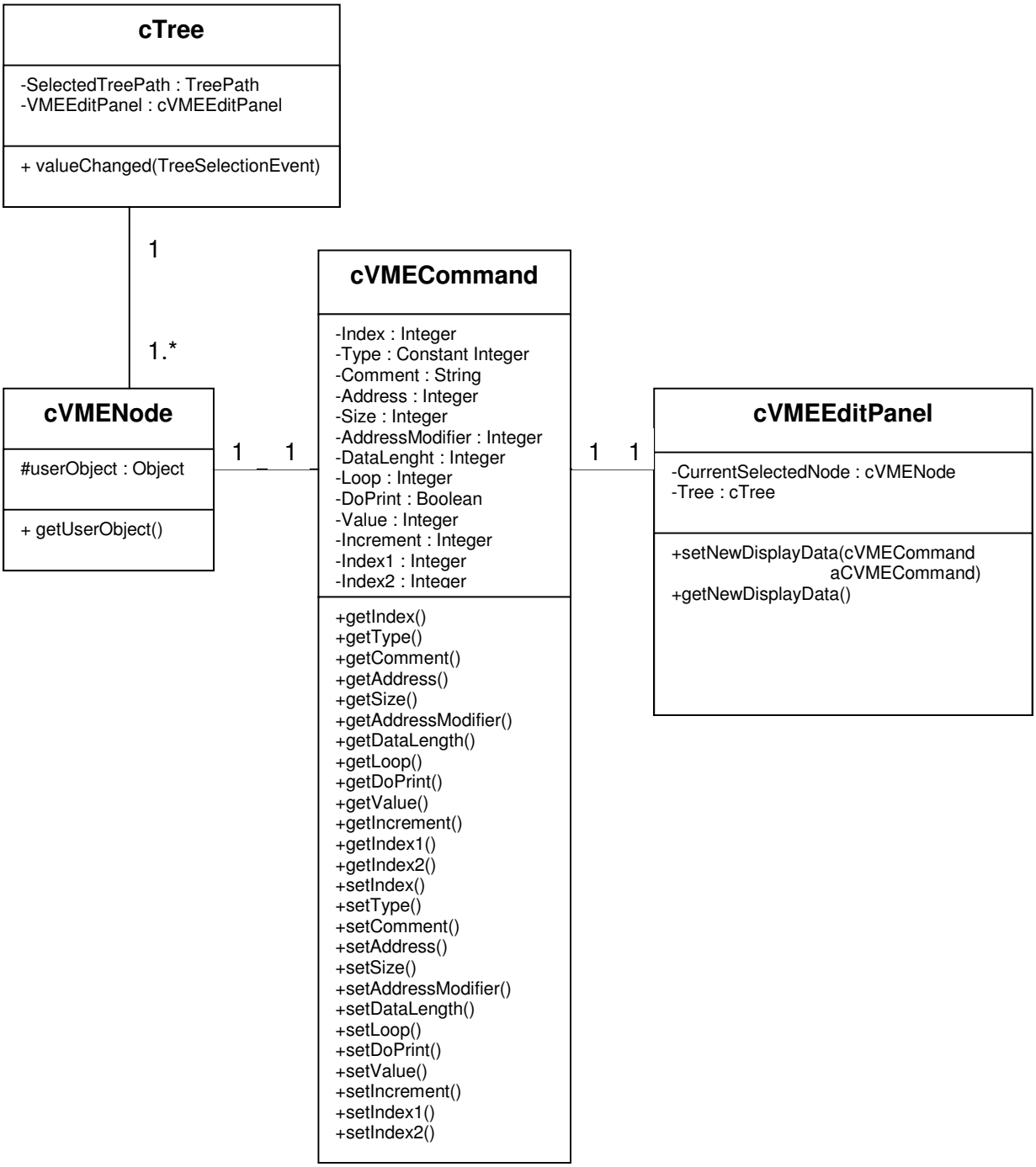
8.4. Erarbeitung der Spezifikation für die Eingabevalidierung

Die Eingabeelemente selbst sollen schon die fehlerfreie Bedienung durch den Benutzer sicherstellen. Daher soll immer nach einer Eingabe durch den Benutzer eine Fehlerprüfung ausgelöst werden, die bei einer fehlerhaften Eingabe das Verlassen des Eingabeelementes nicht zulässt.
In dieser Fehlerprüfung (von hier ab Eingabevalidierung genannt) soll lediglich geprüft werden, ob die Eingaben falsche Zeichen enthalten, oder aber die Eingabe zu lang für den entsprechenden Datentyp ist.
Diese Eingabevalidierung wird auch dazu eingesetzt werden, die Bedienelemente zum Abspeichern und Zurücksetzen der Eingaben zu steuern. Dies wird auch dieses in einer Entscheidungstabelle (Tabelle 3 siehe Anhang) festgehalten.

8.5. Erarbeitung eines Konzeptes zur Integration des Teilprojektes in das Gesamtprojekt

Analysieren und Skizzieren der beteiligten Klassen

Um eine optimale Integration des Teilprojektes zu erreichen, ist eine Analyse der bereits bestehenden Klassen im Gesamtprojekt notwendig, um eine Arbeitsgrundlage für die spätere Implementierung zu schaffen.
Diese Darstellung beeinhaltet nur die am Teilprojekt VMEEditPanel beteiligten Komponenten.



Skizze 2 Die vorhandene Klassenstruktur

Bereits hier wird entworfen, wie die beteiligten Klassen verknüpft werden müssen, um der Hauptaufgabe gerecht zu werden, nämlich dem Transfer der Parameterwerte aus der Klasse cVMECommand in die Klasse cVMEEditPanel. Dies wird mit den Methoden get... und set ... Methoden realisiert, die Parameterwerte aus den Instanzvariablen zurückgeben oder dort setzen.
Zum Setzen der Parameterwerte werden hier die Methoden eingesetzt, die mit set... beginnen, und umgekehrt zum auslesen werden Methoden benutzt die mir get... beginnen.

9. Beschreibung der Vorgehensweise

9.1. Programmierung des Layouts

Um die Eingabeelemente, Bedienelemente und Bezeichnungselemente zu erzeugen und anzuzeigen, ist es nötig, die GUI - Elemente diesem Gridbaglayoutmanager mit Angabe der Parameter für die Positionierung in diesem Raster hinzuzufügen. Zunächst soll aber das Layoutmanagement mit GUI - Elementen, die als Platzhalter dienen, erzeugt und getestet werden, um zu prüfen, ob die Platzierung durch den Layoutmanager korrekt nach der Vorgabe aus der Skizze 1 erfolgt.
Hierfür sollen Bezeichnungselemente als Platzhalter verwendet werden.
Um diesen Test durchführen zu können, sind aber noch Überlegungen darüber anzustellen, wie diese GUI - Elemente später gespeichert werden sollen.

9.1.1. Vorbereitung

In welcher Form sollen die Elemente gespeichert werden ?

Da die grafischen Eingabeelemente und Anzeigeelemente des VMEEditPanels dynamisch in Abhängigkeit vom jeweilig vom Benutzer ausgewählten VME - Befehlstyp ein- oder ausgeblendet werden sollen, ist es erforderlich, Referenzen auf diese Elemente in einer Form zu speichern, die einen Zugriff auf diese Elemente über Schleifenkonstrukte erlauben. Ein solcher Zugriff ist nur dann sinnvoll, wenn für alle Elemente in der Schleife die gleiche Methode aufgerufen wird.
Um dies zu realisieren werden diese Elemente in einem Array gespeichert.
Da aber die Eingabeelemente verschiedener Typen Verwendung finden sollen, ist eine Analyse der Klassenhierarchie der Klassen im Swing - Paket an dieser Stelle notwendig, weil ein Array nur Objekte gleicher Typen speichern kann.
Die Analyse der Klassenhierarchie der im VMEEditPanel verwendeten Elemente ergab, dass Eingabeelemente, Bedienelemente und Bezeichnungselemente (von nun an zusammengefasst als Objekte) von der gleichen Superklasse abgeleitet sind, und dass sie alle über eine Methode verfügen, die für das ein - und ausblenden der betreffenden Elemente genutzt werden kann.
Um eine Speicherung der Objekte in einem Array zu ermöglichen, werden alle Objekte zunächst mit dem Konstruktor der Klasse für das benötigte GUI - Element erzeugt, die Referenz auf dieses Objekt aber für die Speicherung im Array als Datentyp der gemeinsamen Superklasse gespeichert. Dies wird durch ein explizites Casting erreicht.
Diese Form der Speicherung ermöglicht den Aufruf von Methoden der gemeinsamen Superklasse dieser Objekte über ein Schleifenkonstrukt, oder aber nach einem Casting der Referenz im Array auch den Zugriff auf Methoden der Klasse, mit deren Konstruktor das Objekt erzeugt wurde. Das ermittelte Konzept zur Speicherung der Referenzen auf die GUI - Elemente wurde nun als Vorraussetzung für die später anschließende Programmierung des Layoutmanagers in den vorhandenen Programmcode implementiert.

9.1.2. Programmierung der Layoutmanager

Der Layoutmanager wird erzeugt und auf korrekte Funktion geprüft

Um ein korrektes Layoutmanagement sicherzustellen, werden die in der Skizze des Layoutmanagements ermittelten Parameterwerte für den Layoutmanager überprüft. Dies wird für alle Panels durchgeführt, die dem VMEEditPanel (siehe Abb. 2) angehören.

Dies wird folgendermaßen realisiert:

1. Eine Instanz des Layoutmanagers wird erzeugt
2. Dem Layoutmanager werden GUI - Elemente als Platzhalter hinzugefügt
3. Anschließend wird überprüft, ob das Layout funktioniert und die Platzhalter GUI - Elemente korrekt positioniert sind, indem diese Elemente sichtbar gemacht werden.
4. Die Platzhalter GUI - Elemente werden durch die in der Spezifikation (siehe Anhang Tabelle 1) definierten GUI - Elemente ersetzt.

Da nun die korrekte Darstellung durch den Layoutmanager sichergestellt ist, kann mit der Programmierung der eigentlichen Funktionalitäten der Eingabeelemente begonnen werden.

9.2. Programmieren der Eingabeelemente

Eingabeelemente in Java arbeiten mit Ereignissen, die durch eine Aktion vom Benutzer mit diesen Eingabeelementen erzeugt wurden.
Um Eingaben vom Benutzer in diesen Eingabeelementen zu verarbeiten, sind Funktionen nötig, die diese von den Eingabeelementen ausgelösten Ereignisse verarbeiten können. Diese Funktionen stellt Java in Form von Schnittstellen (engl. Interfaces) zur Verfügung. Diese Schnittstellen enthalten eine oder mehrer Methoden, die benannt, aber nicht implementiert sind. Dadurch wird in Java ermöglicht, dass Klassen neben den aus deren Superklassen geerbten Verhaltensweisen noch zusätzliche Verhaltensweisen (Methoden) erhalten, indem in diese leeren Methoden Programmcode zur Verarbeitung dieser Ereignisse eingefügt wird.

Um die Verarbeitung der Eingaben in einem Eingabeelement zu ermöglichen, sind folgende Schritte notwendig:

1. Die benötigten Schnittstellen werden ermittelt und den GUI - Elementen zugeordnet (siehe Tabelle 1). Die erforderlichen Schnittstellen können in den Klassenbeschreibungen der verwendeten Klassen der verwendeten GUI - Elemente ermittelt werden.

GUI - Element	Schaltfläche	Textfeld	Kombinationsfeld	Checkbox
ActionListener Betätigen einer Schaltfläche, Drücken der Returntaste innerhalb eines Testfeldes oder Auswahl eines Menüeintrages	Ja	Nein	Nein	Nein
ItemListener Anklicken einer Checkbox	Nein	Nein	Ja	Ja
FocusListener Ein Ereignis wird ausgelöst, wenn die Komponente den Focus erhält oder verliert	Nein	Ja	Nein	Nein

Tabelle 1 - Zuordnung der benötigten Schnittstellen

2. Einbinden der Schnittstellen als "innere Klassen" der Klasse cVMEEditPanel, damit diese dann die vom Benutzer ausgelösten Ereignisse verarbeiten können.
   Sie werden als "innere Klassen" in die Klasse cVMEEditPanel eingebettet, um den Wartungsaufwand durch diese modulare Programmierweise zu reduzieren.

   Um die Funktionalitäten der Schnittstellen für die GUI - Elemente benutzen zu können, sind folgende Schritte nötig:

   a. Die Schnittstelle wird mit dem Schlüsselwort "*implements"* in der Klassendefinition der in die Klasse VMEEditPanel eingebetteten Klasse mit eingebunden
   b. Es wird eine Instanz der Klasse erzeugt
   c. Diese Instanz wird dem jeweiligen GUI - Element mit dessen entsprechender Methode hinzugefügt.

3. Die in den Schnittstellen deklarierten Methoden können nun von den ihnen zugeordneten GUI - Elementen durch eine vom User ausgelöstes Ereignis aufgerufen werden. In diesen Methoden werden diese Ereignisse verarbeitet und es wird in ihnen gegebenenfalls eine Aktion zur weiteren Verarbeitung ausgelöst. In diesen Methoden werden damit die eigentlichen Funktionalitäten des VMEEditPanels realisiert.

a. MyItemListenerForComboBoxes
Hier wird festgestellt, welchen VME - Befehlstyp der Benutzer gerade ausgewählt hat und damit auch dafür gesorgt, dass die dem entsprechenden VME - Befehlstyp zugeordneten GUI - Elemente ein oder ausgeblendet werden.

b. MyActionListenerForButtonPanel
Hier werden Betätigungen der Schaltflächen verarbeitet.

c. ActionListenerForVMECommandControlPanel
Hier wird festgestellt, ob der Benutzer die Eingabetaste in einem Textfeld betätigt hat, und darauf die anderen GUI - Elemente entsprechend der Gültigkeit der Eingabe in diesem Textfeld gesteuert.
Desweiteren wird hier die Betätigung der Checkbox "Do Print" überwacht und eine Änderung direkt im gerade selektieren VMEbus - Befehls gespeichert.

d. MyFocusListenerForTextFields
Hier wird festgestellt, ob die Eingabeelemente, in denen Texteingaben gemacht wurden, den Eingabefokus verloren haben und dieser gegebenenfalls bei einer ungültigen Eingabe in das Eingabeelement zurückgesetzt.

9.3. Testen der Elemente nach der Programmierung

Dieser Funktionstest soll sicherstellen dass die programmierten GUI - Elemente gemäß der Spezifikation funktionieren. In diesem Test werden systematisch alle Eingabeelemente auf ihre fehlerfreie Funktion überprüft.
Als Arbeitsgrundlage dienen auch hier die in der Spezifikation erstellten Entscheidungstabellen, welche alle zu prüfenden Elemente enthalten.

Hier eine Auflistung der durchgeführten Tests:

Testaufgabe	ggf. Arbeitsgrundlage	Ergebnis
Werden bei einem Wechsel des VMEbus - Befehlstyps nur die dem jeweiligen Befehlstyp zugeordneten Eingabeelemente in dem Panel dargestellt ?	Entscheidungstabelle Tabelle 1	Ja
Führen Fehleingaben in den Eingabeelementen dazu, dass die Schaltfläche zum Abspeichern deaktiviert wird?	Entscheidungstabelle Tabelle 3	Ja
Führen Fehleingaben dazu, dass das Eingabeelement nicht mehr verlassen werden kann?	Entscheidungstabelle Tabelle 3	Ja
Führen richtige Eingaben zu einer Änderung der im VME - Befehl gespeicherten Werte?	Entscheidungstabelle Tabelle 3	Ja
Führt eine Betätigung der Schaltfläche "speichern" zu einer dauerhaften Speicherung?		Ja

Tabelle 2 - Tabelle zur Überprüfung der Funktionen

10. Projektverlauf

1. Tag Projektarbeit 17.05.2002
    1. Erforderliche Datentypen ermittelt
    2. Entscheidungstabelle für GUI - Elemente und Datentypen erstellt
    3. Skizze der GUI - Elemente entworfen
    4. Erforderliche Layoutmanager ermittelt

2. Tag Projektarbeit 20.05.2002
    1. Parameter für Layoutmanager auf Ausdruck der Skizze festgelegt
    2. Erstellen der Skizze der Klassenstruktur

3. Tag Projektarbeit 21.05.2002
    1. Layoutmanagement für GUI - Elemente programmiert und getestet
    2. Entscheidungstabelle für das Anzeigen beziehungsweise das Ausblenden der GUI - Elemente in Arraywerte des Datentyps Boolean übertragen

4. Tag Projektarbeit 22.05.2002
    1. Platzhalter GUI - Elemente durch die in der Spezifikation definierten GUI-Elemente ausgetauscht
    2. ActionListener dem GUI-Element hinzugefügt, welches den VMEbus - Befehlstyp darstellt
    3. Auswahlelemente für den VME -Befehlstyp auf Funktion getestet

5. Tag Projektarbeit 23.05.2002
    1. ActionListener den anderen GUI - Elementen hinzugefügt
    2. Eingabeelemente für die Parameterwerte auf Funktion getestet

6. Tag Projektarbeit 24.05.2002
    1. Programmierung der Eingabevalidierung

7. Tag Projektarbeit 27.05.2002
    1. Programmierung der Eingabevalidierung
    2. Testen der Eingabevalidierung

8. Tag Projektarbeit 28.05.2002
    1. Integration und in das Gesamtprojekt VMEbus - Sequencer
    2. Erstellen der Dokumentation

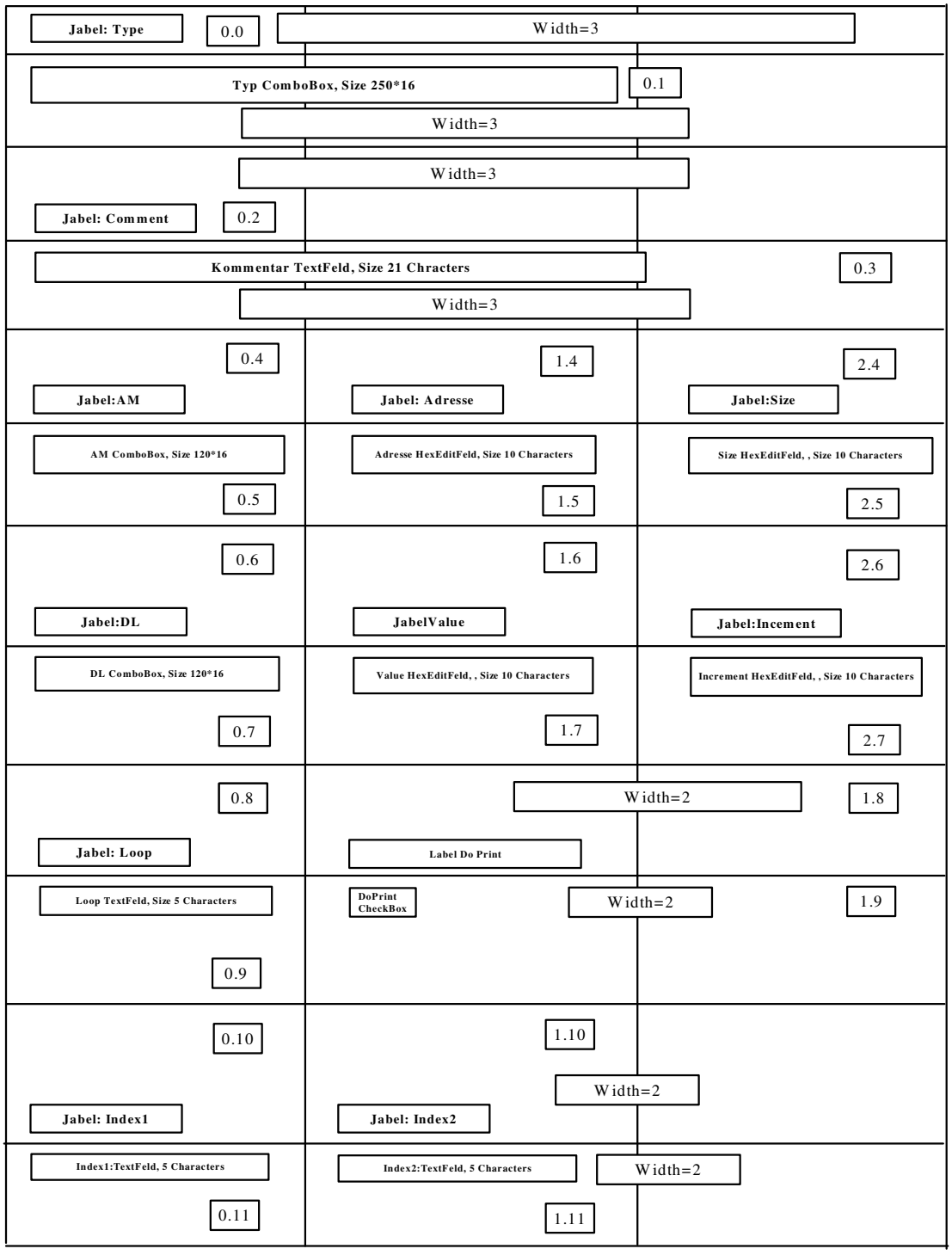
9. Tag Projektarbeit 29.05.2002
    1. Fertigstellen der Dokumentation

11. Glossar

Api:	Applikation Programming Interface
GUI:	Graphical User Interface (zu deutsch: grafische Benutzeroberfläche)
Klasse:	Eine Vorlage für ein Objekt. Diese beeinhaltet Variablen, um das Objekt zu beschreiben, und Methoden, um zu beschreiben, wie sich das Objekt verhält. Klassen können Variablen und Methoden von anderen Klassen erben.
Methode:	Eine Gruppe von Anweisungen in einer Klasse, die definieren, wie sich Objekte dieser Klasse verhalten werden. Methoden sind analog zu Funktionen in anderen Programmiersprachen. Im Unterschied zu Funktionen müssen sich Methoden immer innerhalb einer Klasse befinden.
Objekt:	Eine Instanz einer Klasse. Mehrere Objekte, die Instanzen derselben Klasse sind, haben Zugriff auf dieselben Methoden, aber oft unterschiedliche Werte für deren Instanzvariablen.
Package:	Pakete (engl. Packages) sind eine Möglichkeit, um verwandte Klassen und Schnittstellen zu gruppieren.
Panel:	engl. Bedienendungsfeld, Schalttafel
Schleifenkonstrukt:	Schleifen werden häufig für einfache Wiederholungen verwendet, um Blockanweisungen mehrmals auszuführen und dann zu stoppen.
Schnittstelle:	Eine Beschreibung abstrakter Verhaltensweisen, die einzelne Klassen implementieren können.
Swing:	Swing ist Teil der Java Foundation Classes. Mit Hilfe des Swing Paketes wird die Programmierung grafischer Benutzeroberflächen, wie sie von Microsoft, Linux usw. bekannt sind, wesentlich vereinfacht.

12 Anhang

12.1. Skizze des VMECommandControlPanels



Skizze 1

12.2. Liste vom Auftraggeber

Datenfelder für VME-Kommandos

Allgemein
Index
Typ
Kommentar

Read
Adresse
AM
DL
Loop
DoPrint

Write
Adresse
AM
DL
Loop
DoPrint
Value

Compare
Adresse
AM
DL
Loop
DoPrint
Value
Increment

Read Array
Adresse
Size
AM
DL
Loop
DoPrint

Write Array
Adresse
Size
AM
DL
Loop
DoPrint
Value
Increment

Compare Array
Adresse
Size
AM
DL
Loop
DoPrint
Value
Increment

If Index1 else Index2
Adresse
AM
DL
Loop
DoPrint
Value
Increment
Index1
Index2

GoTo Index
Index

Datentypen / GUI Elemente

Adresse
Integer
HexEditField
Textfeld in das nur Ziffern und die Buchstaben A-F eingetragen werden können. Konvention : Hexzahlen starten mit 0x oder 0X

Size
Integer
HexEditField
Textfeld in das nur Ziffern und die Buchstaben A-F eingetragen werden können. Konvention : Hexzahlen starten mit 0x oder 0X

Value
Integer
HexEditField
Textfeld in das nur Ziffern und die Buchstaben A-F eingetragen werden können. Konvention : Hexzahlen starten mit 0x oder 0X

Inkrement
Integer
HexEditField
Textfeld in das nur Ziffern und die Buchstaben A-F eingetragen werden können. Konvention : Hexzahlen starten mit 0x oder 0X

AddressModifier
constant Integer
A16, A24,A32 + Erweiterungen
ListBox

DataLength
constant Integer
D16, D32
ListBox

Index, Index1, Index2
Integer
Textfeld
Überprüft ob Kommando mit Index schon vorhanden ist (?)

DoPrint
Boolean
Checkbox

Loop
Integer
Textfeld
>=0, 0 entspricht 1

Kommentar
String
Textfeld

12.3. Entscheidungstabellen

Tabelle 1 : Entscheidungstabelle für Datentypen und GUI - Elemente													
Befehlstyp	Index	Type	Comment	Address	Size	AddressModifier	DataLength	Loop	DoPrint	Value	Increment	Index1	Index2
Datentyp	Integer	Constant Integer	String	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer	Integer
Eingabeelement	Textfield	ComboBox	Textfield	Textfield	ComboBox	Textfield	ComboBox	Textfield	Textfield	Textfield	Textfield	Textfield	Textfield

Tabelle 2 : Entscheidungstabelle für die Anzeige der GUI - Elemente								
	Read	Write	Compare	Read Array	Write Array	Compare Array	If Index 1 else Index 2	GoTo Index
GUI - Elemente anzeigen								
Type	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
Comment	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Ja
AM	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein
Address	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein
Size	Nein	Nein	Nein	Ja	Ja	Ja	Nein	Nein
DL	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein
Value	Nein	Ja	Ja	Nein	Ja	Ja	Ja	Nein
Increment	Nein	Nein	Ja	Nein	Ja	Ja	j	Nein
Loop	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein
Do Print	Ja	Ja	Ja	Ja	Ja	Ja	Ja	Nein
Index 1	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein
Index 2	Nein	Nein	Nein	Nein	Nein	Nein	Ja	Nein

Tabelle 3 : Entscheidungstabelle Eingabevalidierung			
Kann Eingabe in dem Eingabeelement zugehörigen Datentyp gespeichert werden?	Ja	Ja	Nein
Entspricht die Eingabe den für diese Eingabeelement definierten Länge?	Ja	Nein	Nein
Enable Button speichern	Ja	Nein	Nein
Enable Button reset	Ja	Ja	Ja