

Entwurfsdokumentation

Visuelle Programmiersprache für den Physikunterricht zur Datenerfassung auf einem Raspberry Pi

Version 0.0.0

David Gawron Stefan Geretschläger Leon Huck
Jan Küblbeck Linus Ruhnke

1. Juli 2019

Inhaltsverzeichnis

1	Ziel der Entwurfsdokumentation	3
2	Klassenbeschreibung	4
2.1	Backend	5
2.2	Model	6
2.3	Controller	7
2.4	View	8
2.4.1	MainWindow	8
2.4.2	ConfigurationField	8
2.4.3	BuildingBlockView	8
2.4.4	SensorBlockView	8
2.4.5	TransformationBlockView	9
2.4.6	RepresentationBlockView	9
2.4.7	PrototypeField	9
2.4.8	Package ButtonLayer	9
2.4.9	Package ExceptionLayer	11
3	Sequenzdiagramme	13
4	Änderungen am Pflichtenheft	14
5	Formale Spezifikationen von Kernkomponenten	15
6	Weitere UML Diagramme	16
7	Anhang	17
7.1	Vollständiges Klassendiagramm	18
8	Glossar	19

1 Ziel der Entwurfsdokumentation

Die Entwurfsdokumentation soll, aufbauend auf das Pflichtenheft, Entwurfsentscheidungen festhalten. Der Rahmen des Entwurfes wird durch einen *Model-View-Controller* (MVC) gebildet. Die Daten werden durch das Backend zu der Verfügung gestellt. Jedes dieser Pakete kommuniziert über eine Fassade. Dadurch werden die Pakete von einander abgekoppelt. Durch diesen grundlegenden Aufbau wird die Software in vier unabhängige Komponenten aufgeteilt, die unabhängig voneinander implementiert und später erweitert werden können.



Abbildung 1: Die grobe Struktur des Entwurfs

2 Klassenbeschreibung

Im folgenden sollen alle Klassen mit ihren Funktion beschrieben werden. Der Aufbau orientiert sich dabei an der in 1 aufgeführten Struktur.

2.1 Backend

2.2 Model

2.3 Controller

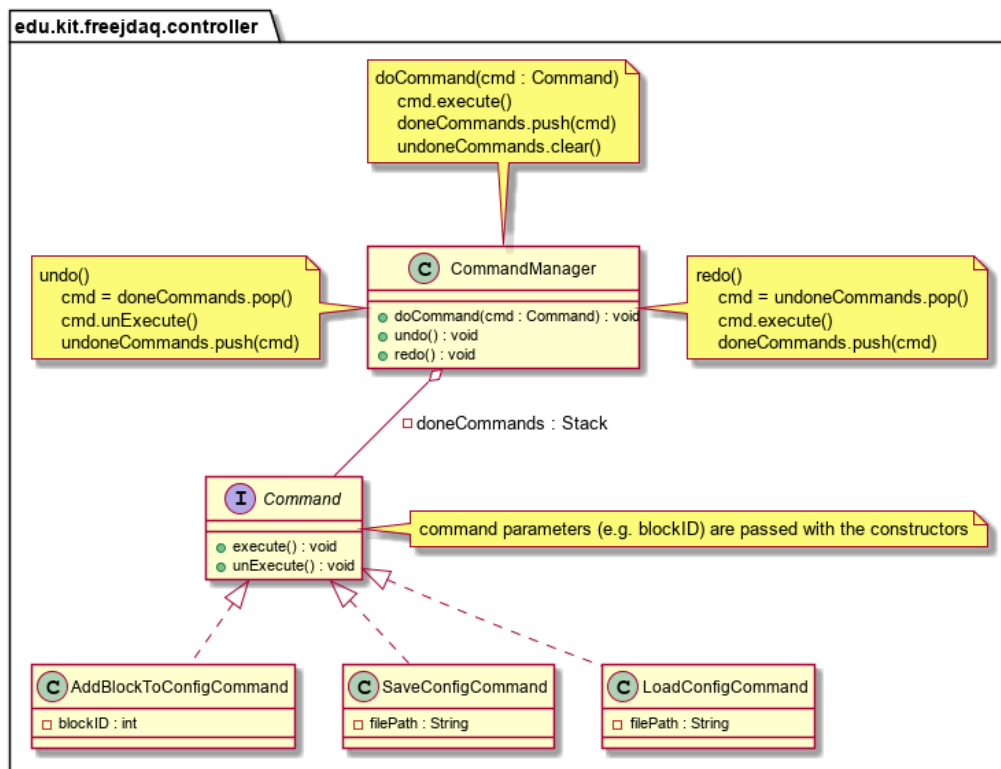


Abbildung 2: Die Struktur des Controllers

2.4 View

Das Paket View, stellt gemäß des MVC- Entwurfsmusters die Darstellungen von Daten des Modells dar und realisiert Benutzerinteraktionen.

2.4.1 MainWindow

Die Klasse MainWindow stellt den Rahmen der Benutzeroberfläche dar. Da MainWindow, dass Entwurfsmuster Singleton verwendet, kann die Anwendung nur ein MainWindow besitzen soll.

2.4.2 ConfigurationField

Die Klasse ConfigurationField stellt das Konfigurationfeld dar, in welchem der Benutzer eine Messkonfiguration aufbauen kann. Konfigurationsbausteine, welche der Benutzer in das Konfigurationfeld platziert werden in einer Liste gespeichert. Konfigurationsbausteine, welche der Benutzer aus dem Konfigurationfeld entfernt, werden aus der Liste gelöscht. Beim Platzieren der Konfigurationsbausteine in das Konfigurationfeld wird dem Konfigurationsbaustein eine eindeutige Position zugeteilt, welche in Form von x- und y-Koordinaten dargestellt wird.

2.4.3 BuildingBlockView

Die Klasse BuildingBlockView ist die Überklasse der Darstellungen der Konfigurationsbausteine. Konfigurationsbausteine besitzen eine eindeutige ID, einen Namen, falls sie im Konfigurationfeld platziert werden ihre Position anhand der Koordinaten x und y. Form und Farbe sind ebenfalls festgelegt.

2.4.4 SensorBlockView

Die Klasse SensorBlockView stellt einen Sensorbaustein dar. Sensorbausteine, welche in dem Konfigurationfeld platziert werden, können mit anderen Bausteinen verbunden werden, was im Messlauf einen Datenfluss über die verbundenen Bausteine erlaubt. Sensorbausteine besitzen, im Gegensatz zu anderen Konfigurationsbausteinen nur Datenausgänge, über welche sie verbunden werden können.

2.4.5 TransformationBlockView

Die Klasse TransformationBlockView stellt einen Transformationsbaustein dar. Transformationsbausteine besitzen eine vordefinierte Funktion, welche die Messdaten verändern.

2.4.6 RepresentationBlockView

Die Klasse RepresentationBlockView stellt einen Darstellungbaustein dar, dieser bestimmt, wie die Messdaten visualisiert werden. - Er besitzt nur Eingänge, aber keine Ausgänge

2.4.7 PrototypeField

Die Klasse PrototypeField ist die Über-Klasse zu SensorBlockField, TransformationBlockField und RepresentationBlockField. Sie stellt eine Fläche dar, in welcher vordefinierte Konfigurationsbausteine dargestellt werden und der Benutzer sie mit dem Mauszeiger in das Konfigurationsfeld ziehen und damit positionieren kann.

2.4.8 Package ButtonLayer

Knöpfe bieten dem Benutzer eine Anzahl von Funktion zur Bedienung der Anwendung an. Das Paket ButtonLayer enthält die unterschiedlichen Knöpfe, das Feld, in welchem die Knöpfe dargestellt werden und eine Annahmestelle für die Benutzerinteraktion.

ButtonField Die Klasse ButtonField stellt das Feld dar, in in welchem die Buttons dargestellt werden. Es enthält also alle konkreten Knöpfe.

Button Die Klasse Button ist die Überklasse zu den konkreten Knöpfen. Jeder Knopf enthält einen eindeutigen Namen und zur Unterscheidung der Knöpfe und zur einfachen Benutzung eine Farbe und ein aussagekräftiges Bild, welches die Funktionalität des Knopfes darstellt.

StartButton Die Klasse StartButton erbt von der Überklasse Button und stellt den Knopf dar, welcher bei Betätigung den Messlauf starten soll.

PauseButton Die Klasse PauseButton ist eine weitere Konkretisierung von Button und stellt den Knopf dar, welcher einen Messlauf pausiert.

ResumeButton Die Klasse ResumeButton stellt den Knopf dar, welcher einen Messlauf fortsetzt.

ResetButton Die Klasse ResetButton stellt den Knopf dar, welcher bei Betätigung den Messlauf auf den Ausgangszustand zurücksetzt.

SaveDataButton Die Klasse SaveDataButton stellt den Knopf dar, welcher dem Benutzer ermöglicht die Messwerte aus einem Messlauf zu speichern.

SaveRepresentationButton Die Klasse SaveRepresentationButton stellt den Knopf dar, welcher eine Momentaufnahme der graphischen Visualisierung der Messwerte speichern lässt.

SaveConfiguration Die Klasse SaveConfiguration stellt den Knopf dar, welcher dem Benutzer erlaubt seine eigene Messkonfiguration zu speichern.

LoadConfiguration Die Klasse LoadConfiguration repräsentiert den Knopf, welcher eine gespeicherte Messkonfiguration in das Konfigurationsfeld laden lässt.

NewConfiguration Die Klasse NewConfiguration stellt den Knopf dar, welcher dem Benutzer die Funktion bietet eine neue Konfiguration zu erstellen.

UndoButton Die Klasse UndoButton stellt den Undo-Knopf dar, welcher bei Betätigung die letzte Benutzeraktion rückgängig macht.

RedoButton Die Klasse RedoButton stellt den Redo-Knopf dar, welcher die letzte rückgängig gemachte Aktion wiederherstellt.

Interface ButtonHandler Das Interface ButtonHandler registriert Benutzerinteraktionen auf der Benutzeroberfläche und löst die Methode ButtonPressed() aus, über welche die Anwendung die Benutzerinteraktion weiterverarbeitet.

2.4.9 Package ExceptionLayer

Fehlernachrichten sind ein wichtiger Teil der Anwendung, um dem Benutzer eine möglichst benutzerfreundliche Umgebung zu liefern und eine möglichst einfache und verständliche Bedienung zu ermöglichen. Damit der Benutzer aussagekräftige Fehlermeldungen erhält unterscheiden wir im Entwurf zwischen drei Typen von Fehlerarten aus verschiedenen Fehlerquellen.

ExceptionWindow Die Klasse ExceptionWindow stellt die Überklasse der drei verschiedenen Unterklassen dar und enthält die gemeinsamen Attribute, welche die konkreten Fehlermeldungen enthalten. Eine Fehlermeldung besitzt immer einen Titel, der wünschenswerter Weise bereits die Fehlermeldung aussagekräftig und kurz beschreibt. Die Beschreibung der Fehlermeldung wiederum liefert eine genauere und explizite Erklärung zur Fehlerquelle, Fehlerursache und möglicherweise ebenfalls zur Fehlerbehebung. Damit der Benutzer auf die Fehlernachricht aufmerksam wird, bewirkt die Methode popUp(), dass die Fehlernachricht zu sehen ist. Damit der Benutzer weiterarbeiten kann oder den Fehler beheben will kann die Fehlernachricht wieder geschlossen werden.

BuildingBlockExceptionWindow Die Klasse BuildingBlockExceptionWindow ist eine Konkretisierung der Überklasse ExceptionWindow und stellt eine Fehlernachricht im Bezug zu Konfigurationsbausteinen dar. Neben einem Titel und einer Beschreibung wird zur Erzeugung dieser Fehlernachricht die eindeutige ID des Konfigurationsbausteins benötigt. Dadurch erfährt der Benutzer sofort, bei welchem Konfigurationsbaustein ein Fehler aufgetreten ist. Die Methode popUp() aus der Überklasse wird hier überschrieben. Damit soll bewirkt werden, dass die Fehlermeldung als Pop-Up Nachricht direkt neben dem Konfigurationsbaustein im Konfigurationsfeld erscheint und somit dem Benutzer sofort die Fehlerquelle signalisiert. Ebenfalls wird zur Darstellung des Fehlers die Farbe des Konfigurationsbausteins im Konfigurationsfeld geändert, um dem Benutzer nochmal auf die Fehlerquelle hinzuweisen.

ConnectionExceptionWindow Die Klasse ConnectionExceptionWindow ist eine weitere Konkretisierung der Überklasse ExceptionWindow und stellt eine Fehlernachricht bei Verbindungen zwischen Konfigurationsbausteinen dar. Zur Identifizierung der Fehlerquelle wird neben Titel und Beschreibung ebenfalls die IDs der Ein- und Ausgangskanäle

der Konfigurationsbaustein mit übergeben. Die Methode `popUp()` soll ebenfalls die Fehlnachricht in der Nähe der Fehlerquelle im Konfigurationsfeld plazieren. Ebenfalls wird die Farbe des Drahtes sinnvoll verändert um die Fehlerquelle zu signalisieren.

GeneralExceptionWindow Die Klasse `GeneralExceptionWindow` stellt neben den zwei konkreten Fehlermeldungen `ConnectionExceptionWindow` und `BuildingBlockExceptionWindow` eine allgemeinere Fehlnachricht dar. Diese werden zum Beispiel bei Messfehlern oder Fehler bei der Messkonfiguration ausgelöst. Diese Fehlnachrichten sollen sichtbar in der Mitte der Anwendung geöffnet werden, um dem Benutzer auf diesen Fehler hinzuweisen.

3 Sequenzdiagramme

4 Änderungen am Pflichtenheft

5 Formale Spezifikationen von Kernkomponenten

6 Weitere UML Diagramme

7 Anhang

7.1 Vollständiges Klassendiagramm

8 Glossar

Model-View-Controller Architekturmuster, dass die Software in die drei Komponenten: Model, View und Controller unterteilt. Dadurch sollen die einzelnen Komponenten unabhängig von einander verändert werden können..