* Pflichtenheft Änderungen
  + Alles Andere
    - …
  + GUI
    - Graph-Preview
      * *Worum geht es?*
        + In der Graph-Preview Ansicht in der GUI werden die einzelnen Graphen, seien sie generiert oder importiert, unter einem neuen Tab angezeigt.
      * *Wie war es bisher?*
        + Diese Anzeige war bisher so gestaltet, dass die Graphen in einer Grid-View gesetzt werden.
        + Dies würde in einer tabellenartigen Darstellung resultieren, bei der Beispielsweise 2 Spalten und 3 Reihen für die Graphen gleichzeitig dargestellt werden.
      * *Was war Grund der Änderung:*
        + Diese Ansicht hatte den Nachteil, dass der User immer gezeichnete Graphen vor sich sieht.
        + Dies führt zu deutlich geringerer Übersichtlichkeit.
        + Außerdem bestand kein großes Interesse des Kunden daran, dass man die zuvor generierten Graphen sofort betrachten kann.

Das graphische Darstellen der Graphen wurde eher an anderer Stelle gewünscht.

* + - * + Darüber hinaus ist diese Art der Ansicht nicht besonders gut skalierbar, wenn der User die Fenstergröße anpassen möchte, besteht die Gefahr, dass die Graphen-Bilder zu klein werden, um anschaulich zu sein.
      * *Was ist die Änderung?*
        + Aus diesem Grund haben wir die Ansicht zu einer Tab-View geändert.
        + Dies bedeutet, dass man nun eine Liste an ausklappbaren Tabs mit den jeweiligen Graphen-Namen vor sich sieht.
        + Demzufolge kann man bei Interesse die Graphen-Tabs ausklappen.

Beim Ausklappen wird dann genau dieser zu betrachtende Graph gezeichnet.

Daraus folgt, dass man nicht mehr mit Graphen-Zeichnungen überschüttet wird.

* + - * *Folgen für das weitere Programm*
        + Durch diese Änderung entsteht ein weiterer Vorteil.
        + Die Performance des Programms wird verbessert, da das Programm nicht sofort alle Graphen zeichen muss, sondern diesen Task erst bei Bedarf starten muss.
    - Generierung
      * *Worum geht es?*
        + Möchte man die Heuristiken anwenden, benötigt man selbstverständlich hierfür erst einmal Graphen.
        + Unser Programm stellt zu diesem Zweck mehrere Beschaffungsmöglichkeiten zur Verfügung:

Automatische zufällige Generierung mit zuvor getätigten Einstellungen.

Import bereits generierter Graphen.

Im Graph-Editor von Grund auf neue Graphen von Hand erstellen.

* + - * *Wie war es bisher?*
        + Unter dem Tab "Graphen Generieren" der GUI war es bisher so gehalten, dass man als erstes die möglichen Einstellungsmöglichkeiten zur Generierung hat und sich darunter dann die verschiedenen Knöpfe befinden, welche die Generierung, den Import, oder das Zeichnen von Hand starten.
      * *Was war Grund der Änderung:*
        + Diese Anordnung macht nur wenig Sinn, da man im Falle eines Imports keine Einstellungsmöglichkeiten benötigt.
      * *Was ist die Änderung?*
        + Aus diesem Grund befinden sich nun die Buttons, welche die einzelnen möglichen Aktionen (Starten der Generierung, des Zeichnens oder des Imports) ausführen, an oberster Stelle.
        + Außerdem werden die Einstellungsmöglichkeiten zur zufälligen Generierung so lange vor dem User verborgen, bis er/sie aktiv auswählt diese Funktionalität wirklich zu benutzen.
* Allgemein
  + Das von uns als Grundstein des gesamten Projekts gewählte Entwurfsmuster ist das „Model-View-Controller“ (MVC) Entwurfsmuster.
  + Es gliedert das Programm, wie der Name schon sehr stark vermuten Lässt in die folgenden drei grundlegenden Bestandteile
    - Modell
      * Das Model beschäftigt sich mit der gesamten Logik des Programms
      * Hierunter zählen in unserem Fall hauptsächlich die verschiedenen Heuristiken.
    - View / Präsentation
      * Die View stellt die Graphische Schnittstelle zum User dar.
    - Controller / Steuerung
      * Der Controller bildet das Zwischenstück zu den bisherigen Teilen und verteilt die Aufgaben.
* Model
  + Dieser Teil wird nicht von mir bearbeitet 😊
* View
  + *Entwicklung*
  + *Einleitung:*
    - Kommen wir nun zum nächsten Großen Abschnitt des Programm-Entwurfs.
    - Nachdem wir im letzten Abschnitt über das Model gesprochen haben folgt nun der View-Teil des "Model-View-Controller"-Entwurfsmusters.
    - Die View beschäftigt sich, wie der Name andeutet mit dem Aussehen des Programms und somit mit der graphischen Repräsentation.
  + *Überleitung:*
    - Wie im Pflichtenheft beschreiben haben wir uns für die Entwicklung mit Java entschieden.
    - Unter Java gibt es mehrere Möglichkeiten eine GUI zu erstellen.
      * + Standart Widget Toolkit (SWT)
        + Abstract Widget Toolkit (AWT)
        + Swing
        + JavaFx
    - Uns war allerdings relativ schnell klar, dass die Wahl auf JavaFx fallen wird.
      * Dies lag nicht zuletzt an FXML und der bisherigen Entwicklungs-Erfahrung.
      * Dazu gleich mehr.
  + Java FX
    - *Allgemeine Beschreibung:*
      * JavaFx ist eine Abkürzung für Java Graphics.
      * JavaFx ist eine Möglichkeit unter Java eine graphische Oberfläche zu erstellen.
      * JavaFx ist eine komplette Neuentwicklung von Oracle.
      * Es ist unabhängig von den bisherigen Methoden AWT und Swing.
      * JavaFx wurde 2014 veröffentlicht.
      * Es ist seit Version 7.6 in x86 Java Standard Edition (JavaSE) Runtime Installation enthalten.
      * Da wir mit Java 8 arbeiten werden ist dies somit kein Problem.
    - Aufbau
      * JavaFX arbeitet mit einem Szenengraphen (engl. scene graph), der die einzelnen Bestandteile einer GUI verwaltet.
      * Auf diesen werden dann alle weiteren Bestandteile gesetzt.
    - FXML
      * *Einleitung:*
        + Wie auch bei den alternativen kann man natürlich auch mit JavaFx über zu schreibenden Code GUI-Objekte erstellen und diese auf den Scenen-Graphen aufbringen.
        + Allerdings besteht mit JavaFx erstmals die Möglichkeit eine neue Form der GUI Entwicklung zu beschreiten.
        + Diese erfolgt in Form von FXML.
      * *Allgemeine Beschreibung:*
        + FXML ist eine deklarative Beschreibung der grafischen Oberflächen auf XML-Basis.
        + Dies bietet einige Vorteile gegenüber der konventionellen GUI-Entwicklung.

Zum einen ist durch diese Technologie die Trennung des Designs der GUI und deren Funktionalität strikt getrennt.

Zum anderen ist das Einfügen von GUI-Bestandteilen, die an mehreren Stellen der Benutzeroberfläche zum Einsatz kommen sehr einfach möglich.

Dies Ermöglicht, dass der mehrfachverwendbare Code nur einmal in einem Separatem FXML-Dokument abgespeichert werden muss und dann über den „include-Tag“ an allen Stellen verwendet werden kann.

* + - * + Darüber hinaus können für die Gestaltung auch Web-Technologien wie CSS eingesetzt werden.

Dies sorgt zusätzlich für eine Trennung von Layout auf der einen und Style und Design auf der anderen Seite, da separate CSS-Dateien erstellt werden können.

Diese können dann in den FXML-Code eingebettet werden, sodass die GUI das Design übernehmen kann.

* + - * *Entwicklung:*
        + Die Entwicklung der FXML-Dateien erfolgt zuerst über den SceneBuilder.
        + Dieser ist ein grafisches Tool, das die Erstellung von FXML-Dateien vereinfacht.
        + Der daraus generierte Code wird bei Bedarf dann nochmals per Hand nachbearbeitet.

Zur Nachbereitung zählen unter anderem auch das Einfügen der „include-Tags“ (wie oben beschrieben).

* + *Entwurf*
  + *Einführung*
    - Der Entwurf der View Gliedert sich prinzipiell in folgendes auf
      * Graphic
      * Drawer
      * Music
  + Graphic
  + Drawer
    - *Beschreibung:*
      * Der Drawer-Package beinhaltet alle nötigen Klassen um den Graph darstellen zu lassen.
      * Dazu gehören zwei Teilpackete
        + DrawerGraph

Beinhaltet lediglich den „Drawer“, also Zeichner des Graphen, der

* + - * + Visualization
    - *Black-Box:*
      * -
    - *Feinentwurf:*
      * TODO
      * Methods
        + Constructors
  + Music
    - *Sollte möglicherweise weggelassen werden??*
    - *Beschreibung:*
      * Dieser Teil ist ein kleines „Easter-Egg“ unsererseits.
      * Es soll dem User ermöglichen während das Programm mit der Generierung oder Berechnung beschäftigt ist Musik zu spielen und somit die Dauer etwas zu überbrücken.
    - *Black-Box:*
      * -
    - *Feinentwurf:*
      * TODO
      * Methods
        + Constructors
* Controller
  + *Einleitung*
    - Dieser Abschnitt beschäftigt sich wie der Titel andeuten lässt mit dem Controller des Projektes.
    - Dieser ist wiederum in zwei Hauptbestandteile unterteilt.
      * Zum einen natürlich den üblichen Controller, zum anderen aber auch einem Graphic-Controller, der sich spezifisch mit dem Controlling der View beschäftigt.
  + Controller
    - Bernard’s Job 😊
  + Graphic Controller
    - *Einleitung:*
    - Der Graphic-Controller oder unter JavaFx üblicherweise auch FxController ist der Teil eines JavaFx-Programms der direkt mit dem von der FXML-Datei bereitgestellten GUI verknüpft ist.
    - Der FxController ist somit ein Separater Teil des Controllers, der sich lediglich mit der GUI beschäftigt und die getätigten Eingaben an die richtigen Stellen im allgemeinen Controller weitergibt.
      * Dies bringt den Vorteil, dass der allgemeine Controller keine Kenntnisse über die GUI benötigt und losgelöst von dieser funktionieren kann.
      * Dadurch ist auch die Modularität in diesem Teil des Entwurfs gewährleistet.
    - *Aufbau:*
* Ressourcen
  + *Einleitung*
    - Die Ressourcen sind alle Dateien, die nicht in direktem Zusammenhang mit der Funktionalität und des Programms stehen und keinen Einfluss auf den Ablauf haben.
    - Hierunter fallen meist Bilder, wie Icons, oder auch andere Mediendateien und vieles mehr.
    - Diese Dateien muss unser Programm aus externen Stellen ziehen.
  + *Entwurf*
    - Diese Daten werden getrennt vom Programmcode abgelegt und dann bei Bedarf aus der vordefinierten Stelle vom Programm eingeladen.
  + *Aufbau*
  + *UML Diagramm der Ressourcen zum Überblick*
  + Resources
    - This contains all the Resources that are needed for the Project.
    - FXML
      * This contains all the FXML files for the GUI.
      * They are arranged in different Sub-Folders to separate.
      * Unterordner
        + TODO
    - Pictures
      * This contains all the Pictures used at the GUI organized by sub-Folders.
      * Icons
        + This contains all Icons for the Buttons, ... of the GUI.
      * Logo
        + This contains all Logos used at the GUI.
    - Music
      * This contains all the Music that can be played by default.
    - StyleSheets
      * This Contains all the CSS-Files for the GUI.
    - Plugins
      * This Contains all the Plugins the User could add to the Rage-Program.
      * By Default, there are the Plugins for the TC and EFL that we should implement.
* Glossar
  + TODO …
  + Firmen / Produkte
    - Sun Microsystems
      * Sun Microsystems ist ein Unternehmen, das 2010 von Oracle aufgekauft wurde.
      * See
        + <https://de.wikipedia.org/wiki/Sun_Microsystems>
    - Oracle
      * Oracle Corporation ist ein US-amerikanischer Soft- und Hardwarehersteller mit Hauptsitz im kalifornischen Redwood City (Silicon Valley). Das Unternehmen ist spezialisiert auf die Entwicklung und Vermarktung von Computer-Hardware und -Software für Unternehmenskunden – insbesondere des Datenbanksystems Oracle Database.
      * See
        + <https://de.wikipedia.org/wiki/Oracle>
        + <https://www.oracle.com/index.html>
    - Java
      * Java ist eine objektorientierte Programmiersprache und eine eingetragene Marke des Unternehmens Sun Microsystems.
      * See
        + <https://de.wikipedia.org/wiki/Java_(Programmiersprache)>
        + <http://www.oracle.com/technetwork/java/index.html>)
    - JavaFx
      * JavaFX (Java Graphics) Framework zur Erstellung plattformübergreifender Java-Applikationen.
    - Standart Widget Toolkit (SWT)
      * …
    - Swing
      * …
    - SceneBuilder
  + Fachbegriffe
    - Graphical User Interface GUI
      * = Graphische Benutzer-Schnittstelle
      * Dies beschreibt all das, was dem User auf dem Monitor angezeigt wird.
  + Entwurfsmuster