## Softwarekonzept\* für "Simple Graph Calculator and Plotter"

*\*) Das ist nur ein Entwurf wie so etwas aussehen könnte.*

### 1. Architektur

**1.1. Architekturmodell**

* **Client-Server-Architektur**: Die Anwendung wird als Standalone-Desktop-Anwendung entwickelt, wobei die gesamte Logik und Datenverarbeitung auf dem Client stattfindet.

**1.2. Schichtenarchitektur**

* **Präsentationsschicht**: Verantwortlich für die Benutzeroberfläche und die Interaktion mit dem Benutzer.
* **Logikschicht**: Beinhaltet die Geschäftslogik für die Berechnung und Visualisierung der Funktionen.
* **Datenzugriffsschicht**: Kümmert sich um das Laden und Speichern der letzten verwendeten Parameter.

### 2. Komponenten

**2.1. Benutzeroberfläche (UI)**

* **Hauptfenster**: Enthält die Hauptansicht mit den Eingabefeldern und dem Plotbereich.
* **Eingabefelder**: Dropdown-Menüs zur Auswahl der Funktion, Textfelder zur Eingabe der Parameter.
* **Plotbereich**: Bereich zur Darstellung der Funktion in 2D.
* **Menüleiste**: Optionen zum Speichern der Grafik und Laden der letzten Parameter.

**2.2. Geschäftslogik**

* **Funktionsmanager**: Klasse zur Verwaltung der verschiedenen Funktionstypen (sin, cos, sinc).
* **Plot-Manager**: Klasse zur Berechnung und Aktualisierung der Plots basierend auf den Eingabeparametern.

**2.3. Datenverwaltung**

* **Parameter-Manager**: Klasse zum Speichern und Laden der letzten verwendeten Parameter in einer lokalen Datei.

### 3. Technologien und Tools

* **Programmiersprache**: C#
* **Framework**: WPF (Windows Presentation Foundation) für die UI-Entwicklung
* **Bibliotheken**:
  + OxyPlot für die Plot-Erstellung
  + Newtonsoft.Json für die Speicherung und das Laden der Parameter

### 4. Design Patterns

**4.1. MVVM (Model-View-ViewModel)**

* **Model**: Daten- und Geschäftslogik (Funktionsmanager, Plot-Manager, Parameter-Manager)
* **View**: Benutzeroberfläche (XAML-Dateien für WPF)
* **ViewModel**: Bindeglied zwischen View und Model, enthält die Logik zur Datenbindung und Befehle für die UI.

**4.2. Singleton Pattern**

* Wird verwendet für den Parameter-Manager, um sicherzustellen, dass nur eine Instanz zur Verwaltung der Parameter existiert.

### 5. User Stories

**User Story 1: Funktionstyp auswählen**

* **Als Benutzer** möchte ich zwischen sin, cos und sinc Funktionen auswählen können, **damit** ich verschiedene Funktionen visualisieren kann.

**User Story 2: Parameter eingeben und Funktion plotten**

* **Als Benutzer** möchte ich Parameter für eine Funktion eingeben können, **damit** ich die entsprechende Funktion im Plotbereich sehen kann.

**User Story 3: Bereich anpassen**

* **Als Benutzer** möchte ich den Darstellungsbereich der Funktion anpassen können, **damit** ich verschiedene Ausschnitte der Funktion sehen kann.

**User Story 4: Letzte Parameter laden**

* **Als Benutzer** möchte ich beim Start der Anwendung die zuletzt verwendeten Parameter sehen, **damit** ich meine Arbeit fortsetzen kann.

**User Story 5: Visualisierung speichern**

* **Als Benutzer** möchte ich die Visualisierung im Vektorformat speichern können, **damit** ich die Ergebnisse für Präsentationen oder Berichte nutzen kann.

### 6. Ablaufdiagramm

**6.1. Hauptablauf**

1. **Anwendung starten**
   * Laden der letzten verwendeten Parameter aus der lokalen Datei
2. **Funktion auswählen**
   * **Auswahl der gewünschten Funktion aus einem Dropdown-Menü**
3. **Parameter eingeben**
   * Eingabe der gewünschten Parameter in die entsprechenden Felder
4. **Plot aktualisieren**
   * Echtzeit-Aktualisierung des Plots basierend auf den Eingaben
5. **Bereich anpassen**
   * Anpassung des Darstellungsbereichs durch Eingabe neuer Werte
6. **Visualisierung speichern**
   * Speichern der aktuellen Visualisierung im Vektorformat (SVG, PDF)

### 7. Sequenzdiagramm

Benutzer -> UI: Starten der Anwendung

UI -> Parameter-Manager: Letzte Parameter laden

Parameter-Manager -> UI: Parameter bereitstellen

Benutzer -> UI: Funktionstyp auswählen

Benutzer -> UI: Parameter eingeben

UI -> Funktionsmanager: Parameter übergeben

Funktionsmanager -> Plot-Manager: Funktion berechnen

Plot-Manager -> UI: Plot aktualisieren

Benutzer -> UI: Bereich anpassen

UI -> Plot-Manager: Bereichsparameter übergeben

Plot-Manager -> UI: Plot aktualisieren

Benutzer -> UI: Visualisierung speichern

UI -> Plot-Manager: Grafikdaten anfordern

Plot-Manager -> UI: Grafikdaten bereitstellen

UI -> Dateisystem: Grafikdaten speichern

### 8. Datenmodell

**Parameter**

* Typ: string (sin, cos, sinc)
* Bereich: (double, double)

### 9. Validierung und Verifizierung

**9.1. Testfälle**

* **Parameter-Eingabe**: Testen der Eingabefelder mit verschiedenen Parametern
* **Funktionstyp-Auswahl**: Testen der Dropdown-Auswahl für verschiedene Funktionstypen
* **Plot-Aktualisierung**: Testen der Echtzeit-Aktualisierung des Plots bei Änderung der Parameter
* **Bereich-Anpassung**: Testen der Anpassung des Darstellungsbereichs
* **Speichern und Laden**: Testen der Speicher- und Ladefunktionalität für die Parameter und die Visualisierung

**9.2. Benutzerakzeptanztests**

* Durchführung von Tests mit einer ausgewählten Benutzergruppe zur Validierung der Anforderungen und Benutzerfreundlichkeit

Dieses Softwarekonzept verwendet bewährte Methoden wie die Schichtenarchitektur und das MVVM-Pattern, um eine strukturierte und leicht erweiterbare Anwendung zu gewährleisten. Die klare Trennung von Verantwortlichkeiten und die detaillierte Planung der Komponenten und Abläufe stellen sicher, dass die Anwendung den Anforderungen entspricht und effizient entwickelt werden kann.