Sudoku  
數獨

SWE – Projekt

Projektentwicklung mit Scrum

Autoren

Roman Brülisauer

Christoph Capiaghi

Sedric Christopher

Jasmin Injodikaran

Samuel Moos

Rico Pauli

Abgabedatum

26. Mai. 2017

Zusammenfassung

Die Zusammenfassung enthält die wichtigsten Ergebnisse und Zahlen.

* Fragestellung
* Methode
* Ergebnisse
* Keine Bilder in der Zusammenfassung
* Umfang: Maximal eine Seite

Inhaltsverzeichnis

[1 Einleitung 4](#_Toc481474659)

[1.1 Rahmenbedingungen 4](#_Toc481474660)

[2 Arbeitskonzept 5](#_Toc481474661)

[3 Verwendetes Programmmuster 5](#_Toc481474662)

[4 Struktur der SW 5](#_Toc481474663)

[5 Lösungsalgorithmus 5](#_Toc481474664)

[5.1 Back-Tracking 5](#_Toc481474665)

[5.2 Logic Solver 6](#_Toc481474666)

[5.3 Kombi 6](#_Toc481474667)

[5.4 Gewählter Algorithmus 6](#_Toc481474668)

[6 Benutzerdokumentation 6](#_Toc481474669)

[7 Zeitplan 6](#_Toc481474670)

[8 Schlussfolgerungen / Ausblick 7](#_Toc481474671)

[9 Literaturverzeichnis 8](#_Toc481474672)

Anhangsverzeichnis

[Anhang A Produkteanforderungen 10](#_Toc481474673)

[Anhang B Projektmanagement 11](#_Toc481474674)

[Anhang C Berechnungen 12](#_Toc481474675)

[Anhang D Simulationen 13](#_Toc481474676)

[Anhang E Schema 14](#_Toc481474677)

[Anhang F Layout 15](#_Toc481474678)

[Anhang G Stückliste mit Kostenschätzung 16](#_Toc481474679)

Verzeichnis der Bilder

Bild 1‑1 … 4

Bild 1‑1 … 5

Bild 1‑1 … 6

Verzeichnis der Tabellen

Tabelle 1‑1: Zeitplan Sudoku 7

# Einleitung

Aufgabenstellung

**Die Ausgangssituation:**

Aufgabenstellung ist die Entwicklung eines Sudoku Spiels.

**Rahmenbedingungen die vorgegeben waren:**

**Beschreibung der Lösung:**

Unsere Lösung bietet die vordefinierten Rahmenbedingungen und es wurden zum anderen verschiedene Löser erstellt und getestet.

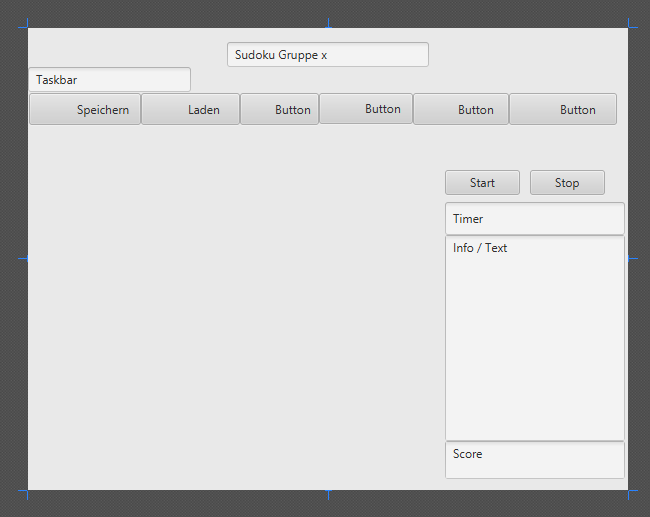


Bild ‑ …

Über die Buttons können die jeweiligen Aktion ausgeführt werden.

Das Spiel lädt bei starten das vorhanden File ein um die vorgegebene Spielgrösse zu erstellen.

## Rahmenbedingungen

* Das Spiel muss definiert werden können z.B. als JASON-File, welche die Spielfeldgrösse, die Felder und evtl. die Lösung definieren.
* Es soll ein Spiel eingelesen und angezeigt werden können. Die Lösung soll mitten im Spielverlauf angezeigt werden können.
* Nach dem Ersten Klicken eines Feldes soll die Zeitmessung starten.
* Die Eingaben sollen Rückgängig gemacht werden können.
* Die Eingaben können jederzeit auf Fehler hin untersucht werden und angezeigt werden. Die Falschen stellen sollten rot dargestellt werden.
* Das Spiel kann jederzeit neu gestartet werden.
* Der Aktuelle Zustand des Spiels kann gespeichert werden
* Ein Automatischer Löser soll implementiert werden.
* Es soll die Möglichkeit geben Zahlen in Hilfsfelder zu notieren.
* Das Spiel soll per Maus oder Tastatur bedient werden können.

## Spielfeld Definition

Das Spielfeld ist intern als 2-dimensionales Byte Array mit jeweils 9 Elementen abgelegt. Die Initialisierung ist wie folgt:

|  |
| --- |
| byte[][] field = new byte[9][9]; |

Leere Felder sind mit der Zahl Null (0) aufgefüllt.

# Arbeitskonzept

* SCRUM Prozess
* git hub
* Rollenverteilung (scrum master)

Dies soll anhand des Scrum Vorgehens ausgeführt. Den Überblick und die Entwicklungsschritte werden in unserem Team via GitHub immer auf dem Laufenden gehalten.

* <https://github.com/Injodikaran/Sudoku/projects/1>

# Struktur der SW

* Klassendiagramm
* Use Case Diagramm

## Speicher Konzept

Zum Speichern und Laden der Spielfelder der wurde das JSON[[1]](#footnote-1)-Dateiformat ausgewählt. Pro Spiel wird ein eigenes File erstellt mit folgenden Inhalt:

* Als erstes wird der Zwischenstand aufgelistet («Sudoku», als byte[][])
* Danach folgt das Template, also das leere Spielfeld («Template» als byte[][])
* Am Schluss folgt der Timer («Time» als String)

Damit sind alle benötigten Parameter abgespeichert.

Beispiel eines gespeicherten Spiels:

|  |
| --- |
| **{**  "Sudoku"**:** **[**  **[**5**,** 3**,** 0**,** 0**,** 7**,** 0**,** 0**,** 0**,** 0**],**  **[**6**,** 0**,** 0**,** 1**,** 9**,** 5**,** 0**,** 0**,** 0**],**  **[**0**,** 9**,** 8**,** 0**,** 0**,** 0**,** 0**,** 6**,** 0**],**  **[**8**,** 0**,** 0**,** 0**,** 6**,** 0**,** 0**,** 0**,** 3**],**  **[**4**,** 0**,** 0**,** 8**,** 0**,** 3**,** 0**,** 0**,** 1**],**  **[**7**,** 0**,** 0**,** 0**,** 2**,** 0**,** 0**,** 0**,** 6**],**  **[**0**,** 6**,** 0**,** 0**,** 0**,** 0**,** 2**,** 8**,** 0**],**  **[**0**,** 0**,** 0**,** 4**,** 1**,** 9**,** 0**,** 0**,** 5**],**  **[**0**,** 0**,** 0**,** 0**,** 8**,** 0**,** 0**,** 7**,** 9**]**  **],**  "Template"**:** **[**  **[**5**,** 3**,** 4**,** 6**,** 7**,** 8**,** 9**,** 1**,** 2**],**  **[**6**,** 7**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 8**],**  **[**1**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 7**],**  **[**8**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 3**],**  **[**4**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 1**],**  **[**7**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 6**],**  **[**9**,** 6**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**],**  **[**2**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 0**,** 5**],**  **[**3**,** 4**,** 5**,** 2**,** 8**,** 6**,** 1**,** 7**,** 9**]**  **],**  "Time"**:** "1.00 s"  **}** |

## Lösungsalgorithmus

Für das automatisierte Lösen eines Sudokus gibt es mehrere Ansätze. Zur Lösungssuche wurden verschieden Algorithmen implementiert und getestet. Nachfolgend werden verschiedene Konzepte beschrieben und die Vor- und Nachteile aufgezeigt.

### Back-Tracking

Ein Ansatz war, einen reinen Back-Tracking Algorithmus zu implementieren. Da die Anzahl möglicher Lösungen begrenzt ist und mit heutiger Hardware kein Problem ist diese zu druchsuchen, ist diese Art von Lösungssuche möglich (brute force). Es wurde unabhängig voneinander zwei Back-Tracking Lösungen implementiert. Diese wurden auf Performance getestet.

Grundsätzlicher Ablauf des Algorithmus:

* Zuerst wird eine Kopie vom Eingabefeld angelegt, damit das originale Feld nicht verändert wird. Dieser Teil des Programms wird nur einmal aufgerufen
* Danach folgt der Teil, welcher wiederholt wird (Funktion BacktrackingAlgo):
  + Abbruchbedingung prüfen: y-Koordinate > 9? → Return true
  + Falls keine Zahl im originalen Feld vorgegeben ist: Setze die Zahl 1 ein und überprüfe, ob die Eingabe gültig war
    - Falls die Zahl gültig war: Setzte sie eine, gehe auf das nächste freie Feld und rufe rekursiv die Funktion BacktrackingAlgo mit den neuen Koordinaten auf.
      * Falls die Funktion true als Rückgabewert gibt: Return true
      * Falls die Funktion false als Rückgabewert gibt: Lösche die Zahl.
    - Falls die Zahl ungültig war: Lösche die zuvor eingesetzte Zahl
  + Dies wird für die Zahlen 1- 9 wiederholt. Falls keine Zahl eingesetzt werden kann: Return false
  + Falls bereits eine Zahl vorgegeben war: Springe zum nächsten Feld.

Vorteile

* Der Algorithmus findet bei einem gültigen Feld immer eine Lösung

Nachteile

* Langsamer im Vergleich von «intelligenten» Lösern

### Logic Solver

Der Logic Solver zählt die Anzahl gültigen Zahlen (1-9) pro Feld. Ist nur noch eine Zahl möglich, setzt er diese ein. Somit löst der Algorithmus das Sudoku gleich wie ein menschlicher Spieler.

Vorteile

* Falls man logische Zwischenschritte anzeigen möchte, eignet sich dieser Algorithmus. In dieser Arbeit wurde auf diese Option verzichtet.

Nachteile

* Falls eine Annahme gemacht werden muss, findet der Algorithmus keine Lösung.

### Kombi

Algo von Samuel

### Gewählter Algorithmus

In dieser Arbeit wurde der kombinierte Algorithmus gewählt, da er positive Eigenschaften aus beiden Algorithmen vereint. Es wird immer eine Lösung gefunden.

## GUI

Muster:

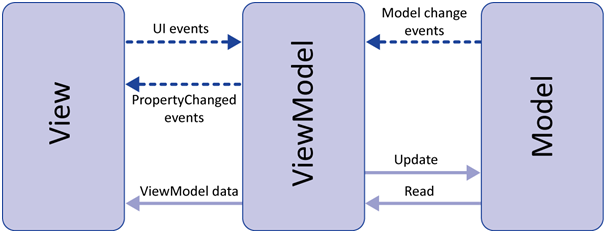


Bild ‑ …

https://msdn.microsoft.com/en-us/library/ff798384.aspx

MVVM wobei bei Javafx das Muster speziell ergänzt (Controller) wird, um die Beziehungen zu realisieren.

# Benutzerdokumentation

Hardware / Software:

Geschwindigkeit:

Speichergrösse:

Datentransport in der Software:

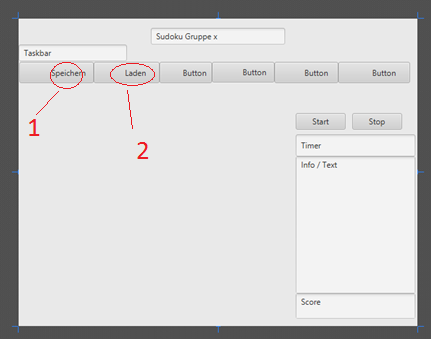


Bild ‑ …

# Grober Zeitplan

Eine Grobe Zeit Übersicht um die Abläufe der jeweiligen Prozesse im Überblick zu haben.

Tabelle ‑: Zeitplan Sudoku

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Zeitplan Projekt Sudoku 2017 | | | | | | | | | | |
|  | 28.02 | 7.03 | 14.03 | 21.03 | 28.03 | 4.04 | 11.04 | 28.04 | 26.05 | 6.06 |
| Arbeitsauftrag erhalten: |  |  |  |  |  |  |  |  |  |  |
| Teambildung: |  |  |  |  |  |  |  |  |  |  |
| GitHub einrichten: |  |  |  |  |  |  |  |  |  |  |
| Arbeitsaufteilung: |  |  |  |  |  |  |  |  |  |  |
| Arbeitsaufträge erledigen |  |  |  |  |  |  |  |  |  |  |
| Komponente überarbeiten: |  |  |  |  |  |  |  |  |  |  |
| Komponente zusammenfügen: |  |  |  |  |  |  |  |  |  |  |
| Dokumentation zusammenstellen: |  |  |  |  |  |  |  |  |  |  |
| Präsentation zusammenstellen: |  |  |  |  |  |  |  |  |  |  |
| Abgabe: |  |  |  |  |  |  |  |  |  |  |
| Präsentation: |  |  |  |  |  |  |  |  |  |  |

# Schlussfolgerungen / Ausblick

* Ausbaufähig
* Schwierigkeitsstufen

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | D. H. Phan, S. Bonnet, R. Guillemaud, E. Castelli and N. Y. Pham Thi, „ Estimation of respiratory waveform and heart rate using an acceleromoter,“ in *Proc. 30th Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society EMBS 2008*, 2008, pp. 4916-4919. |
|  |  |
|  |  |
|  |  |

Anhang

1. Produkteanforderungen

* F: Festanforderung, muss erfüllt werden
* M: Min / Max-Forderung, muss erfüllt werden
* W: Wunschforderung

1. Projektmanagement

* Zeitplanung
* Protokolle (falls vorhanden)
* Budget (falls vorhanden)

1. JSON: JavaScript Object Notation, Datenformat [↑](#footnote-ref-1)