Designdokument

Sudoku-Löser

Abgabe am: 19.06.2018

Autor	Sebastian Bernauer
Kurs	TINF16B5
Studiengang	Angewandte Informatik an der Dualen Hochschule Baden-Württemberg Karlsruhe

Inhaltsverzeichnis

Wichtiger Hinweis

Das Modul der Bilderkennung ist aus Github (https://github.com/prajwalkr/SnapSudoku) entnommen und wurde nur leicht angepasst. Es ist nicht meine Arbeit und soll nicht in die Bewertung mit einfließen, sondern ist nur ein kleiner Zusatz zur besseren Verwendung des fertigen Programms. Es ist auch nicht sonderlich präzise und könnte mit zusätzlichen Daten für eine bessere Genauigkeit trainiert werden.

Einführung und Ziele

Dieses Designheft wird für einen Sudoku-Löser erstellt.

Es soll die Aufgaben und Architektur des Sudoku-Lösers darstellen.

Aufgabenstellung

Die Anforderungen für den Sudoku-Löser sind dem Anforderungsdokument in der Version 1.0 zu entnehmen.

Der Sudoku-Löser soll in erster Linie Sudokus lösen. Dafür soll er folgende Use-Cases unterstützen:

1. Der Nutzer gibt zunächst sein Sudoku ein
2. Der Nutzer wählt aus folgenden Optionen:
    a. Der Nutzer lässt sich sein komplettes Sudoku lösen
    b. Der Nutzer markiert mittels Doppelklick Felder, welche er gelöst haben möchte. Anschließend werden diese Felder von dem Sudoku-Löser gelöst.
    c. Der Nutzer möchte alle Felder markiert haben, welche direkt berechenbar sind
    (Direkt berechenbar heißt in diesem Fall, dass das Feld mittels einer gängigen Strategie in Sudoku in einem Schritt errechnet werden kann)
3. Das Ergebnis wird angezeigt

Qualitätsziele

Absteigend nach Priorität sortiert

Qualitätsziel	**Kurze Beschreibung**
Korrektheit der Lösung	Das Sudoku soll korrekt gelöst werden
Kurze Laufzeiten	Das Lösen des Sudokus sollte maximal 15s in Anspruch nehmen.

Stakeholder

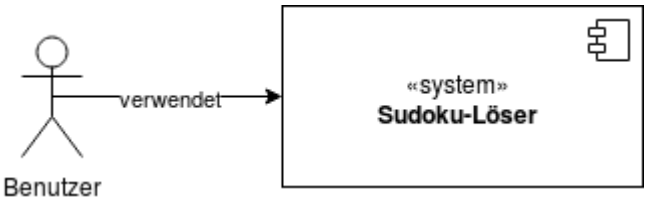
Rolle	Kontakt	Erwartungshaltung
Kunde / Abnehmer	Alexander Kosik	Das Projekt soll die Anforderungen aus dem Anforderungsdokument erfüllen. Das Projekt muss bis zu der Abgabefrist fertig sein.
Nutzer		Der Nutzer möchte ein einfach zu bediendenes Program, welches einwandfrei funktioniert. Der eigentliche Softwareerstellungsprozess ist ihm egal.

Randbedingungen

Der Sudoku-Löser soll plattformübergreifend (Linux, Windows, MacOS) laufen
Der Sudoku-Löser soll in Java implementiert werden
Der Sudoku-Löser sollte auf normalen Computern laufen (4GB RAM, i3, Display mit mind. HD-Auflösung)
Der Sudoku-Löser soll lokal, d.h. ohne Internetanbindung funktionieren

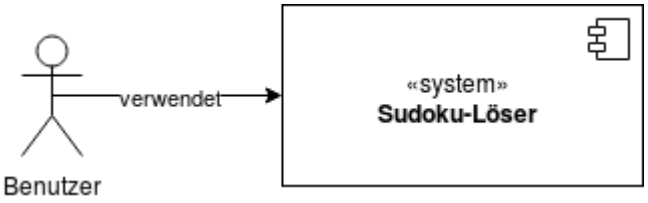
Kontextabgrenzung

Fachlicher Kontext



Es gibt nur den Benutzer und den Sudoku-Löser, weitere Personen / Komponenten sind nicht beteiligt.

Technischer Kontext



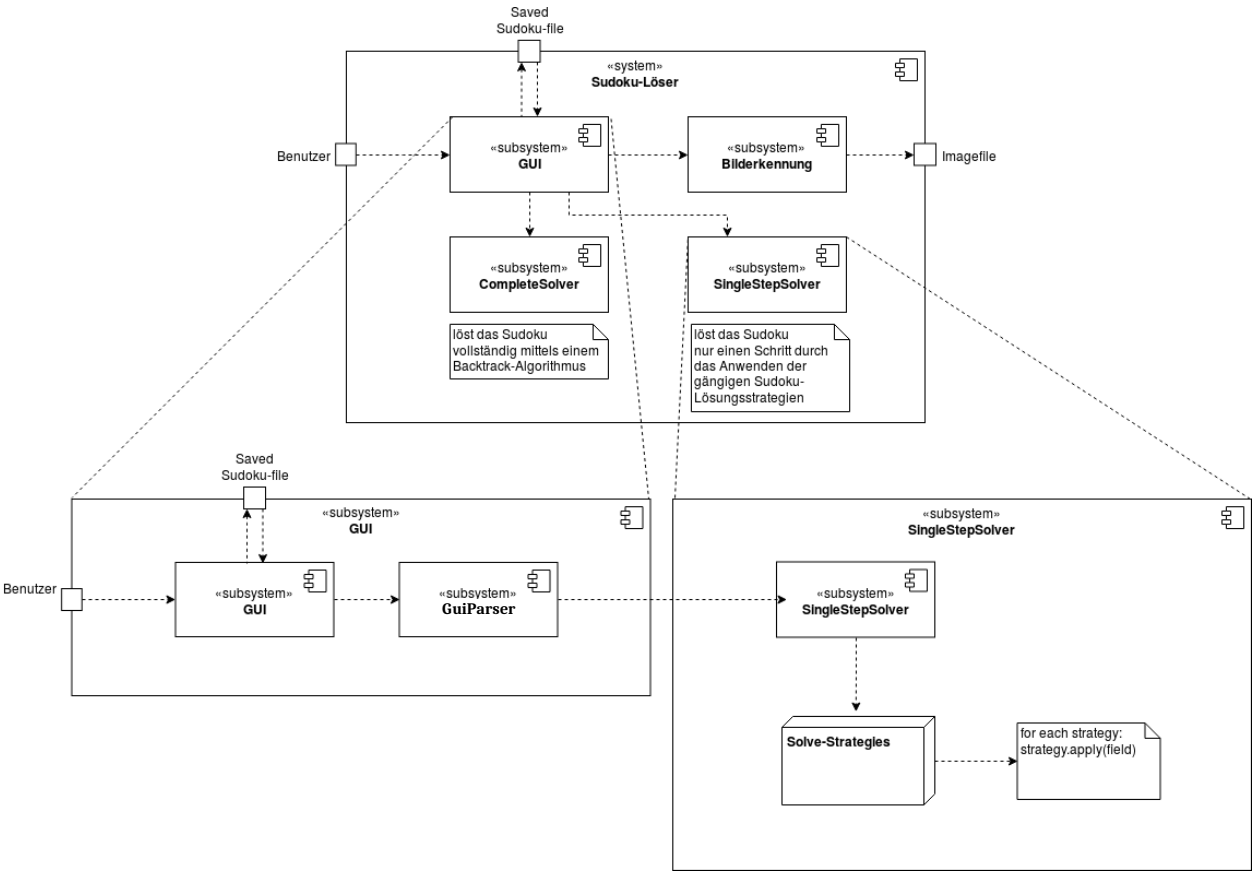
Wie bei dem fachlichen Kontext sind keine weitere Personen / Komponenten beteiligt.

Lösungsstrategie

- Das Projekt wird in einem Maven-Projekt verwaltet. Das erleichtert die Testausführung und das Bauen der ausführbaren .jar-Datei.
- Es gibt einen Solver (BackTrackSolver), welcher das Sudoku komplett löst
- Es gibt einen Solver (SingleStepSolver), welcher nur den Durchlauf aller Strategien einen Schritt weit durchführt.
  Dadurch wird das Sudoku nur einen Schritt gelöst.

  => Die letzen beiden Punkte werden aufgrund der komplett anderen Vorgehensweise festgelegt. Der komplette Solver arbeitet mittels einem Backtrack-Algorithmus, während der SingleStepSolver nur mithilfe der Strategien arbeitet.

Bausteinsicht

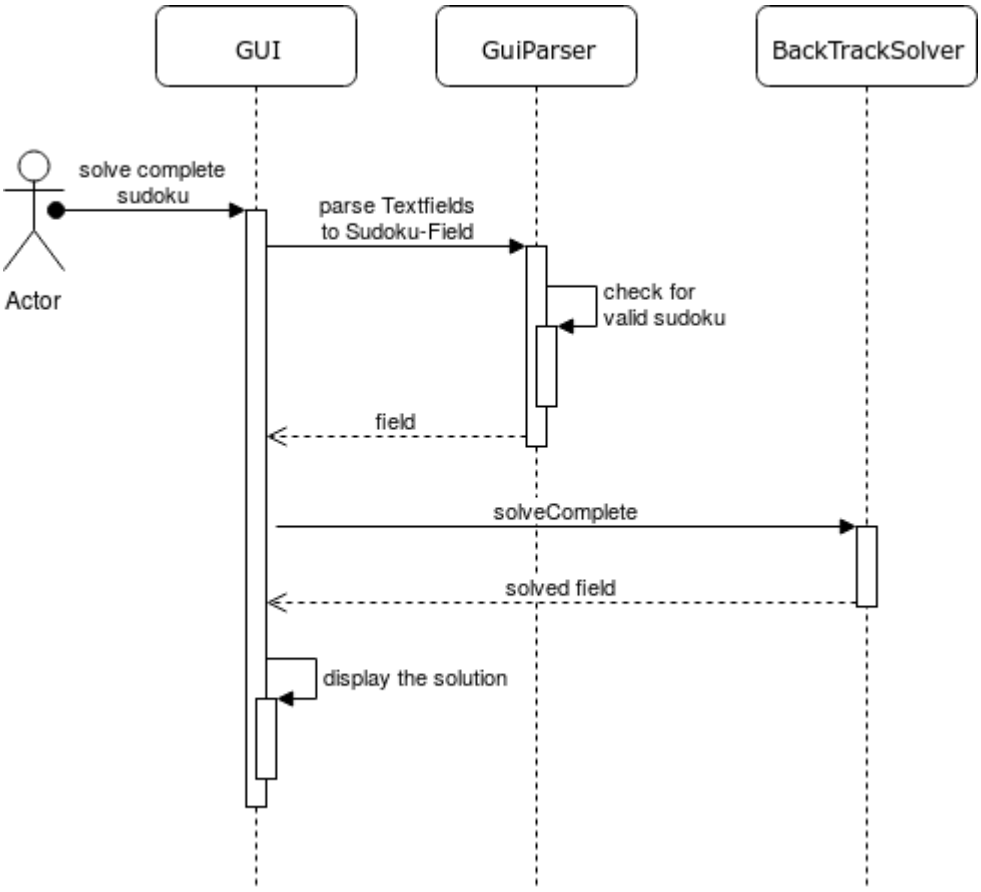


Der GuiParser nimmt die Eingaben in Form eine TextFeld-Arrays entgegen und wandelt dieses in ein Integer-Array um.

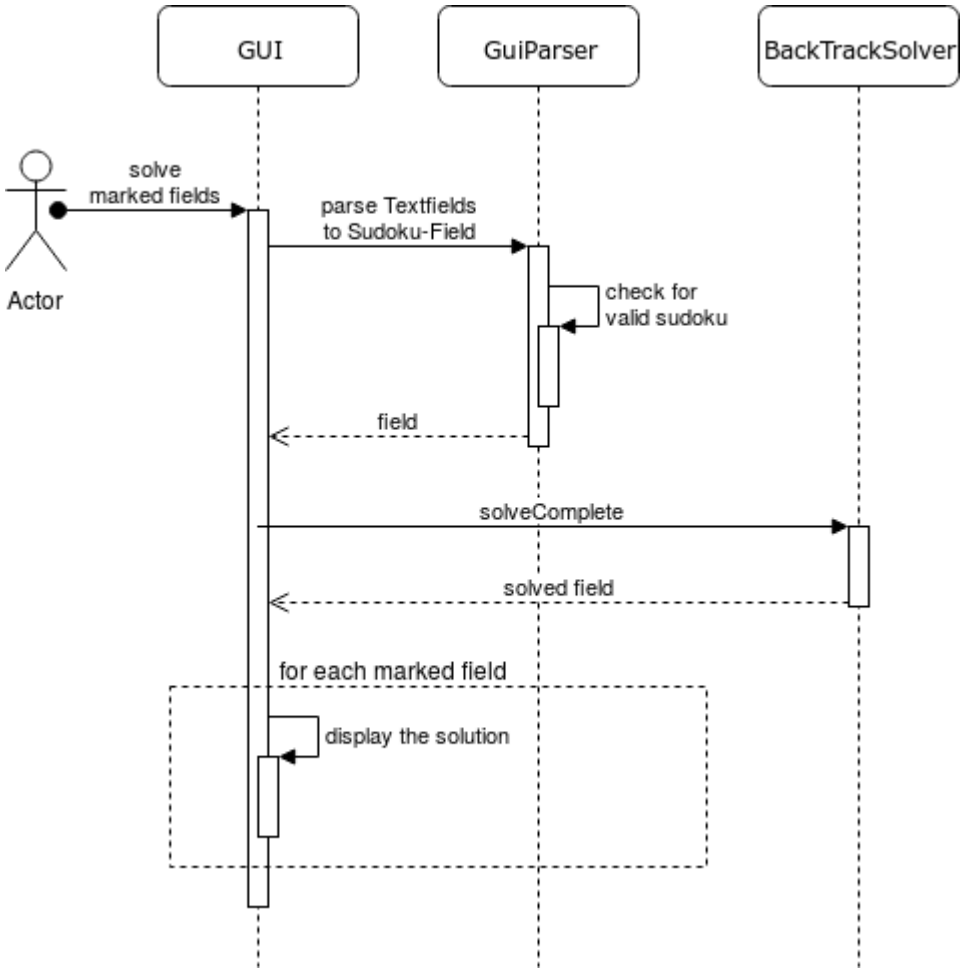
Die Bilderkennung nimmt ein Bild von einem Sudoku entgegen und wandelt dieses in ein Zahlen-Array um. Das Modul wurde aus Github entnommen und nur geringfügig angepasst. Deshalb wird es nicht weiter als Whitebox zerlegt. Es ist in Python geschrieben und verwenden maschinelles Lernen zum Erkennen des Sudokus und der Ziffern.

Laufzeitsicht

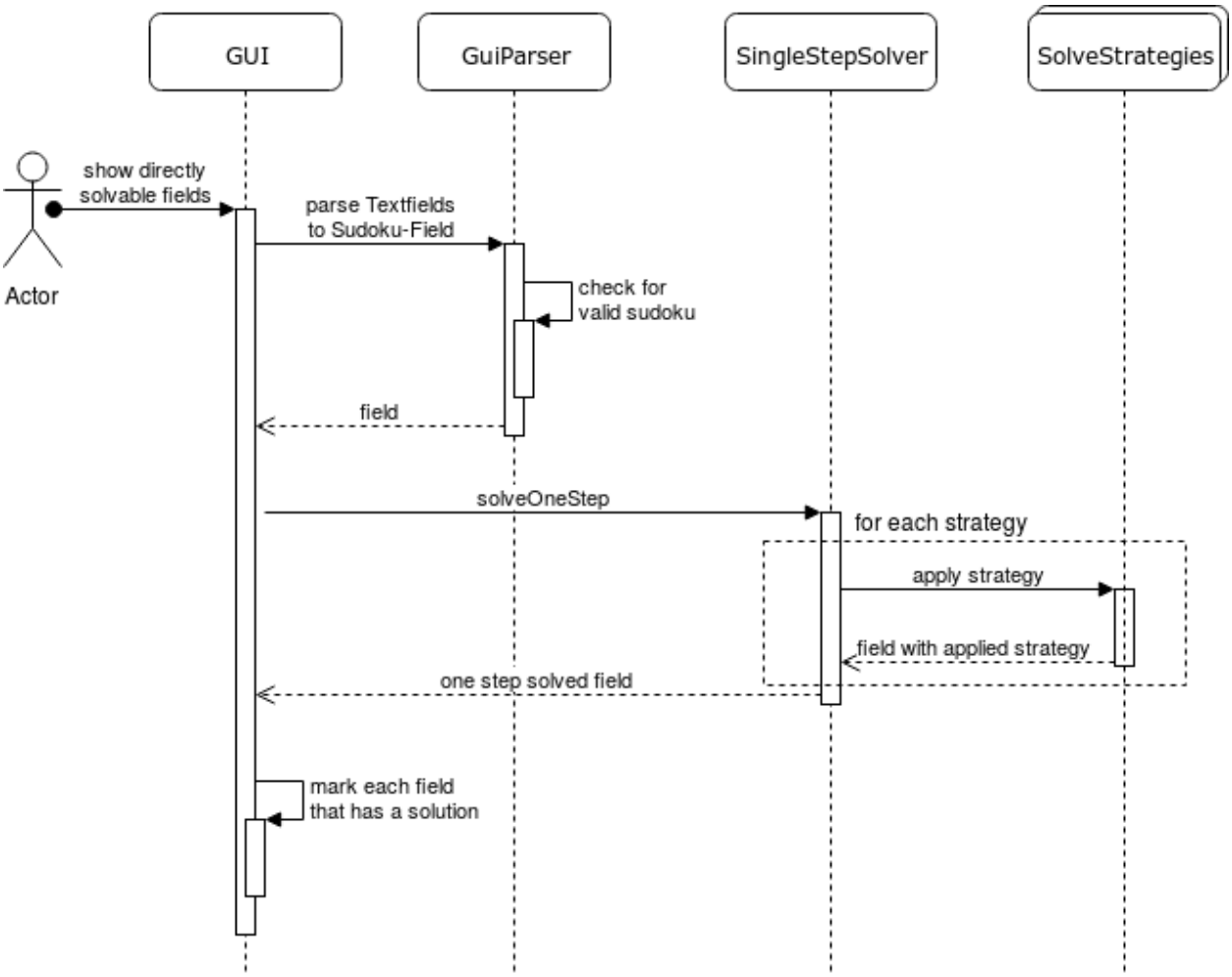
Use-Case: Das gesamte Sudoku soll gelöst werden



Use-Case: Nur die markierten Felder des Sudokus sollen gelöst werden

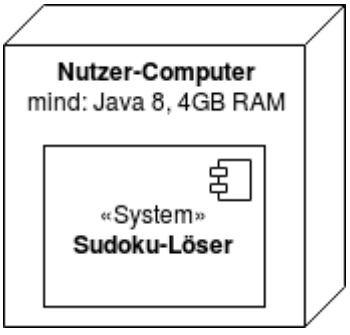


Use-Case: Es sollen alle direkt berechenbaren Felder des Sudokus markiert werden



Verteilungssicht

Der Sudoku-Löser wird auf dem Computer ausgeführt. Es sind keine weiteren Rechner beteiligt.



Entwurfsentscheidungen

Es gibt ein Interface „SolveStrategie“. Damit können alle Strategien gleich behandelt werden. Auch ist es problemlos möglich eine Lösungsstrategie hnzuzufügen, hierfür muss kein Code geändert werden, es muss lediglich eine neue Klasse geschrieben werden, welche das Interface „SolveStrategie“ implementiert.

Das geht in Einklang mit dem Open Close Principle .

Risiken und technische Schulden

Der BackTrack-Solver - der das komplette Sudoku löst – kann unter sehr seltenen Umständen bis zu 30s brauchen. Allerdings ist dies ein eher theoretischer Ansatz.

In der Praxis wurden Tests durchgeführt und es wurde kein Lösungsvorgang gefunden, der länger als 200ms benötigte.