Inhaltsverzeichnis

[Einführung und Ziele 1](#_Toc514217771)

[Aufgabenstellung 1](#_Toc514217772)

[Qualitätsziele 2](#_Toc514217773)

[Stakeholder 2](#_Toc514217774)

[Randbedingungen 2](#_Toc514217775)

[Kontextabgrenzung 3](#_Toc514217776)

[Lösungsstrategie 3](#_Toc514217777)

[Bausteinsicht 4](#_Toc514217778)

[Laufzeitsicht 5](#_Toc514217779)

[Verteilungssicht 8](#_Toc514217780)

[Querschnittliche Konzepte 8](#_Toc514217781)

[*<Konzept 1>* 8](#_Toc514217782)

[*<Konzept 2>* 8](#_Toc514217783)

[*<Konzept n>* 8](#_Toc514217784)

[Entwurfsentscheidungen 8](#_Toc514217785)

[Risiken und technische Schulden 8](#_Toc514217786)

[Glossar 8](#_Toc514217787)

# 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. Zustätzlich soll er folgende Use-Cases ünterstützen:

1. Der Nutzer gibt zunächst sein Sudoku ein
2. Der Nutzer wählt aus folgenden Optionen:
   1. Der Nutzer lässt sch sein komplettes Sudoku lösen
   2. Der Nutzer markiert mittels Doppelklick Felder, welche er gelöst haben möchte
   3. 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 direct 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. |

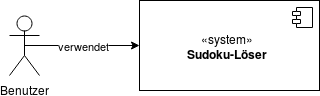
## Stakeholder

|  |  |  |
| --- | --- | --- |
| 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 |

# Randbedingungen

|  |
| --- |
| Der Sudoku-Löser soll platformübergreifend (Linux, Windows, MacOS) laufen |
| Der Sudoku-Löser soll dafür 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 funkionieren |

# Kontextabgrenzung

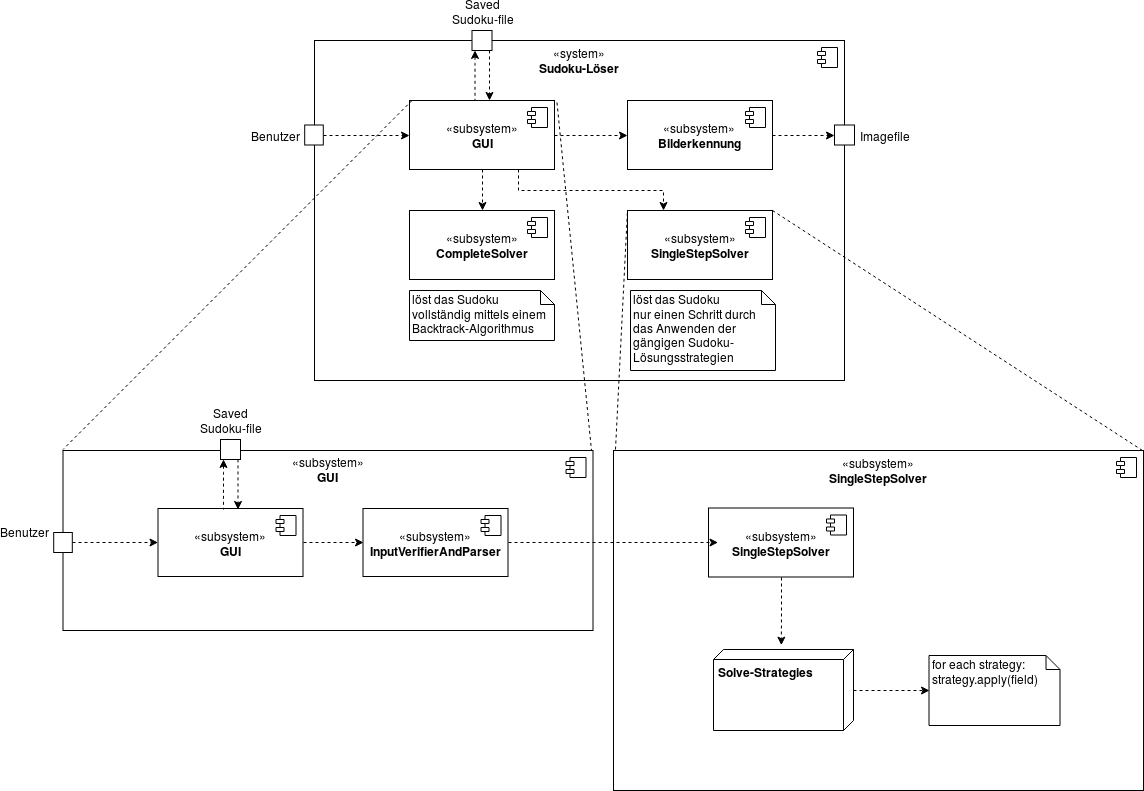


# Lösungsstrategie

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

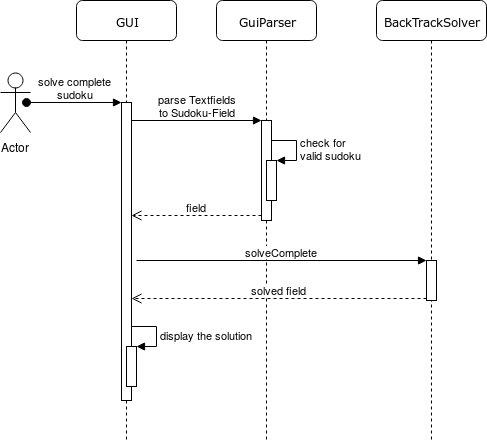
=> Die letzen beiden Punkte werden aufgrund der komplett anderen Vorgehensweise festgelegt. Der komplette Solver arbeitet mittels einem Backtrack-Algorithmus, während der SingleStepSolver nur mithilfe der Strategien arbeitet.

# Bausteinsicht

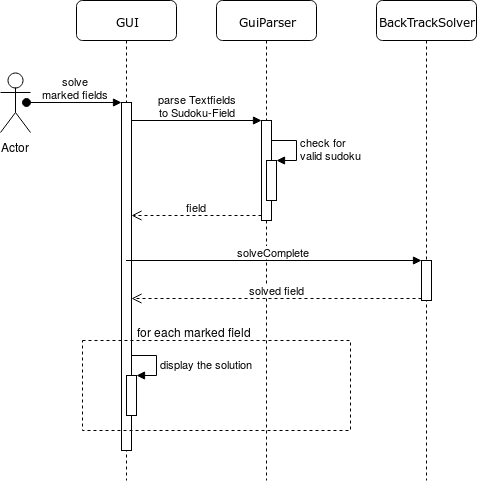


# Laufzeitsicht

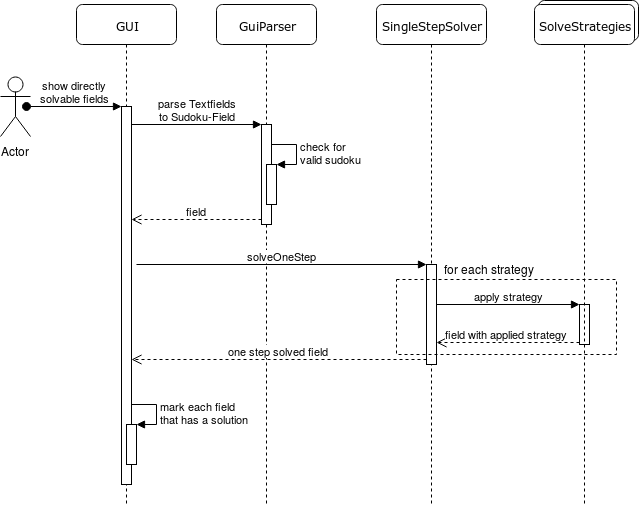
**Use-Case: Das gesamte Sudoku soll gelöst werden**



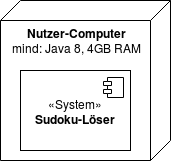
**Use-Case: Nur die markierten Felder des Sudokus sollen gelöst werden**



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



# Verteilungssicht



# Querschnittliche Konzepte

## *<Konzept 1>*

*<Erklärung>*

## *<Konzept 2>*

*<Erklärung>*

…

## *<Konzept n>*

*<Erklärung>*

# Entwurfsentscheidungen

# 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 hat sich gezeigt, dass er normalerweise nie über 100ms benötigt.

# Glossar

|  |  |
| --- | --- |
| Begriff | Definition |
| *<Begriff-1>* | *<Definition-1>* |
| *<Begriff-2* | *<Definition-2>* |