oose.
Innovative Informatik

Vortrag:

Matt in drei Iterationen.

Lebendiger Architekturentwurf am Beispiel einer Schach-Engine

Stefan Zörner (Stefan.Zoerner@oose.de)

Berlin, den 29. März 2012
Berlin Expert Days

BED BERLIN EXPERT DAYS
{join the talk}

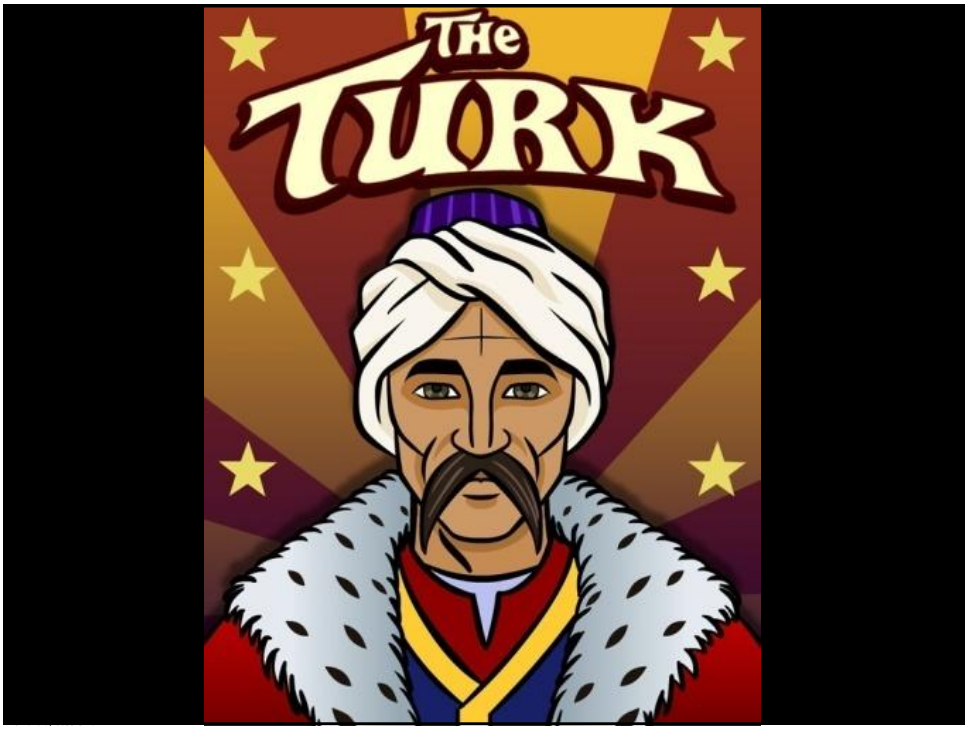
Matt in drei Iterationen
Lebendiger Entwurf am Beispiel einer Schach-Engine

Ein Jahrhunderttraum wie das Fliegen: Eine Maschine, die Menschen im Schach bezwingt. Auch heute für viele Java-Entwickler noch eine faszinierende Aufgabe! Wie zerlegt man das Problem geschickt? Welche wichtigen Entscheidungen sind bei der Umsetzung zu treffen?

Schritt für Schritt entsteht eine spielfähige Schach-Engine. Nach jeder Iteration messen wir anhand einer Live-Partie den Fortschritt. Ihr erhaltet in diesem Vortrag nützliche Tipps für den Start und lernt eigentlich alles, um selbst ein Schachprogramm in Java bauen zu können. Und Ihr erfahrt auf vergnügliche Weise nebenbei, wie Ihr ganz allgemein eine nachvollziehbare, angemessene Softwarearchitektur entwerfen, bewerten und festhalten könnt. En passant.

Zielgruppe

Zielgruppe dieses Vortrags sind in erster Softwareentwickler und -architekten, die neugierig sind, wie eine Schach-Engine funktioniert. Und die anhand dieses Beispiels ein wenig über Architekturentwurf erfahren wollen. Fundierte Schachkenntnisse sind nicht erforderlich.

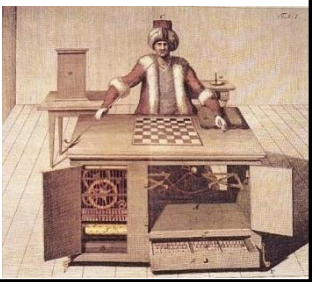


Der Schachtürke (Wolfgang von Kempelen, 1769)

”Meine Damen und Herren, ich habe eine Maschine gebaut wie es sie bisher noch nie gegeben hat: Einen automatischen Schachspieler! Er ist in der Lage, jeden Herausforderer zu schlagen … ”

(von Kempelen, zu Beginn jeder Vorführung)


MAELZEL'S EXHIBITION,
No. 20, St. James's Street.
The Automaton Chess Player
AUTOMATON TRUMPETER,
Conflagration of Moscow,



Schach …

Matt in drei Iterationen.
oose. Innovative Informatik

Claude E. Shannon (1916 – 2001)

” Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance.”

(aus “Programming a computer to play chess”, 1949)

- amerikanischer Mathematiker, Kryptologe, …
- Begründer der Informationstheorie
- Bahnbrechend für Computerschach: „Programming a computer to play chess”

Stefan Zörner …

… bei oose seit 2006 als Trainer und Berater

… regelmäßig Trainings und Workshops zu:

- Softwareentwurf und -architektur, insbesondere Architekturdokumentation
- Umsetzung mit Java-Technologien

… war auf der Suche nach einem lebendigen Beispiel für

- Entwurfsprinzipien und –muster
- Architekturentwurf und vor allem: Architekturdokumentation

- **Fasziniert vom Thema**
- **neugierig, wie aufwendig eine eigene Umsetzung tatsächlich wäre**

... selbst mäßiger Gelegenheitsschachspieler

Matt in drei Iterationen.
oose. Innovative Informatik

Mission Statement – 2 Ziele für den Vortrag

1

Ihr erfahrt die Antwort auf die Frage:
Wie schreibe ich eine eigene Schachengine?

Ihr seid am Ende mit dem Wissen ausgestattet, selbst eine zu schreiben,

bzw. Ihr könnt abschätzen, wie aufwendig das wäre.

(Spaß-Teil)

2

Ihr erfahrt nebenbei etwas über
Architekturentwurf, -bewertung, -dokumentation

Wir hacken nicht einfach drauf los, sondern gehen methodisch vor.

(Ernst-Teil)

Agenda

1. Die Aufgabe
2. Iteration 1: „Der Durchstich“
3. Iteration 2: „Das Bauerndiplom“
4. Iteration 3: „Der Taktikfuchs“
5. Ausblick und Weitere Informationen

Matt in drei Iterationen.
oose. Innovative Informatik

Agenda

1. Die Aufgabe
2. Iteration 1: „Der Durchstich“
3. Iteration 2: „Das Bauerndiplom“
4. Iteration 3: „Der Taktikfuchs“
5. Ausblick und Weitere Informationen

1

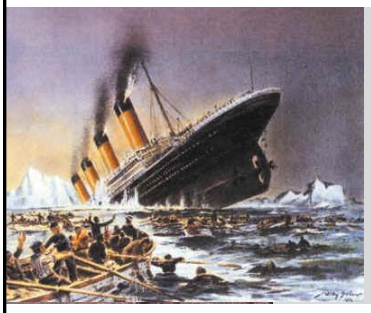
„DokChess“ – Ziele und Features

- DokChess ist eine voll funktionsfähige Schachengine
- Sie dient als einfach zugängliches und zugleich ungemein attraktives Fallbeispiel für Architekturentwurf , -bewertung und -dokumentation.
- Der verständliche Aufbau lädt zum Experimentieren und zum Erweitern der Engine ein
- Ziel ist nicht die höchstmögliche Spielstärke – dennoch gelingen Partien, die Gelegenheitsspielern Freude bereiten.

Wesentliche Features

- Vollständige Implementierung der FIDE-Schachregeln
- Unterstützt das Spiel gegen menschliche Gegner und andere Schachengines
- Beherrschung zentraler taktischer Ideen, beispielsweise Gabel und Spieß
- Integration mit modernen graphischen Schach-Frontends

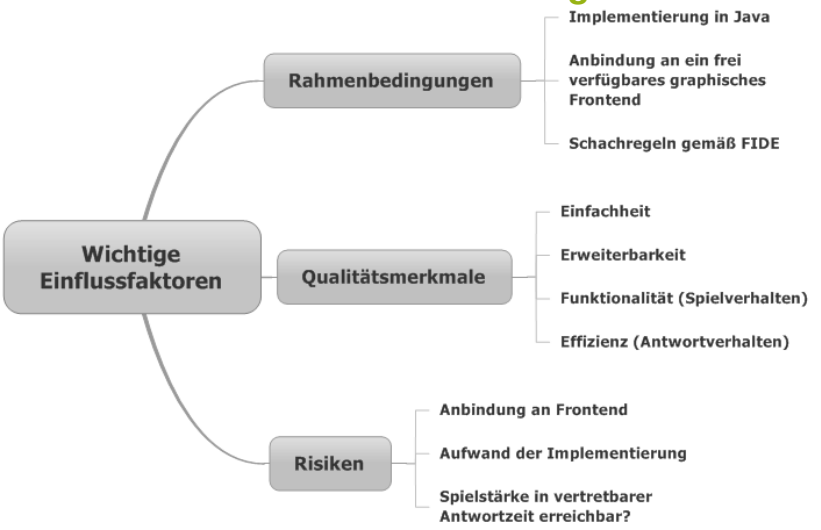
Softwarearchitektur. Eine (!) Definition



“Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.” (Eoin Woods)

- Architekturentscheidungen sind diejenigen, die sich im weiteren Verlauf nur sehr schwer revidieren lassen.
- Konsequenzen: höhere Kosten, Zeitverlust, ggf. scheitert das Vorhaben

Einflussfaktoren auf Entscheidungen



Matt in drei Iterationen.

Jetzt: Drei Iterationen

1 2 3

Gleichförmiger Aufbau in der Darstellung

- Zu Beginn: Vorstellung des Iterationszieles
- Darstellung zentraler Konzepte, Entscheidungen
- Tipps und Tricks
- Am Ende: Spiel gegen die Engine, Bewertung

Agenda

1. Die Aufgabe
2. Iteration 1: „Der Durchstich“
3. Iteration 2: „Das Bauerndiplom“
4. Iteration 3: „Der Taktikfuchs“
5. Ausblick und Weitere Informationen

2

Matt in drei Iterationen.
oose. Innovative Informatik

1. Iteration („Durchstich“), Steckbrief

Ziel:

Engine interagiert mit menschlichem Spieler über ein graphisches Frontend. Es entwickelt sich eine “Partie” über mehrere Züge.

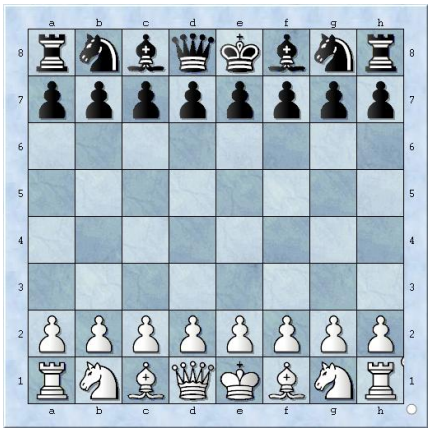
Zentrale Entscheidungen:

- Darstellung der Spielsituation („Stellung“)
- Auswahl eines graphischen Frontends

Implementierungsaufgaben

- Darstellung des „Brettes“, Felder, Züge, etc.
- Anbindung an das graphische Frontend
- Trivialer Zuggenerator

Die Schachdomäne



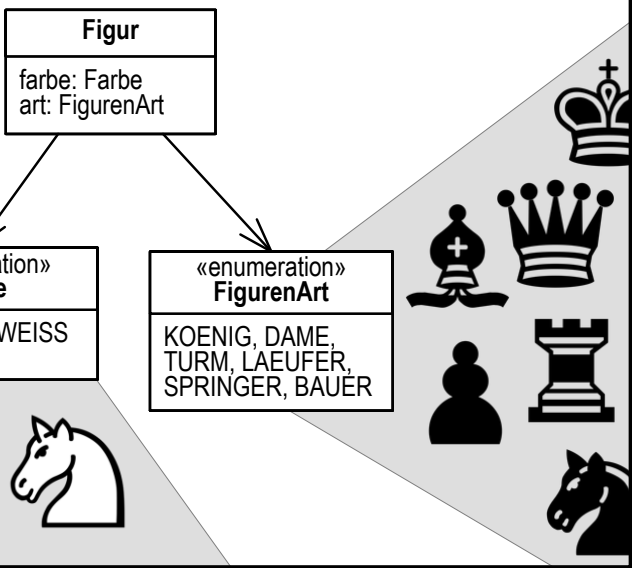
„Das Schachspiel wird zwischen zwei Gegnern gespielt, die abwechselnd ihre Figuren auf einem quadratischen Spielbrett, Schachbrett genannt, ziehen.“
FIDE-Regeln

- Schachbrett 8 x 8 Felder
- 8 Reihen (1-8) und 8 Linien (a-h)
- 2 Spieler, Farben: Schwarz und Weiß
- Figurenarten: König, Dame, Turm, Läufer, Springer, Bauer

- Gezogen wird von Feld zu Feld, gegnerische Figuren werden „geschlagen“
- Ziel: den gegnerischen König zu fangen („Schach matt“)

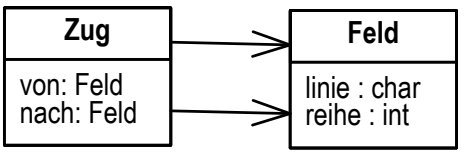
Matt in drei Iterationen.
oose. Innovative Informatik

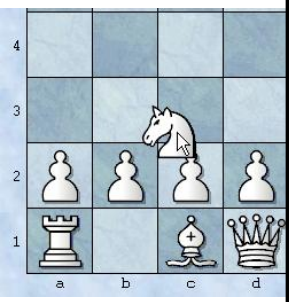
Einfache Darstellung von Figuren



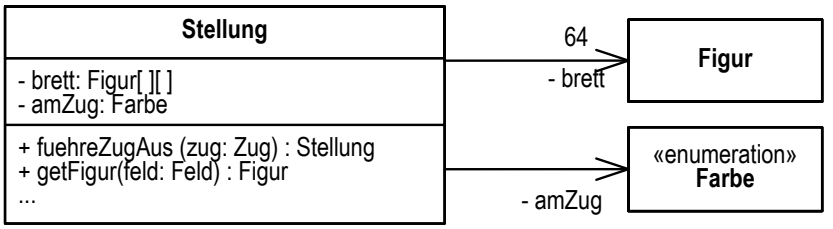
Züge und Felder als Klassen

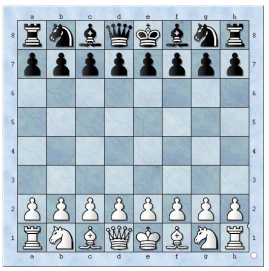
b1 – c3



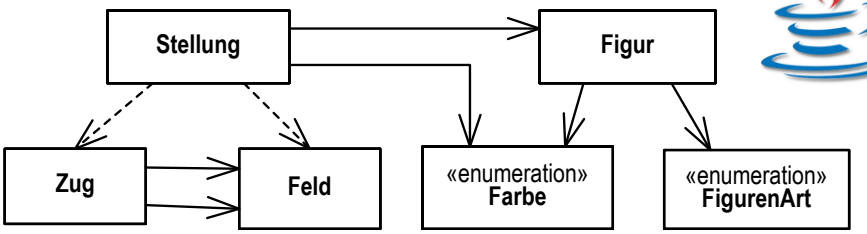


Einfache Darstellung der Stellung





Implementierung in Java



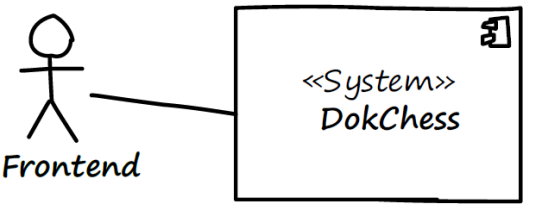
- 4 Klassen
- 2 Aufzählungstypen
- Unit-Tests für `Stellung`

Stellung.java

```
/**
 * Beschreibt eine Spielsituation. Hierzu zaehlen die Figuren auf dem
 * Brett(Stellung), die Farbe am Zug, Rochaderechte usw.
 *
 * @author StefanZ
 */
public class Stellung {

    private Farbe amZug;

    private Figur[][] brett;

    public Stellung() {
        this.amZug = Farbe.WEISS;
        this.brett = new Figur[8][8];
```


Matt in drei Iterationen.
oose. Innovative Informatik

Auswahl + Anbindung graphisches Frontend



Zentrale Anforderungen an Frontend:

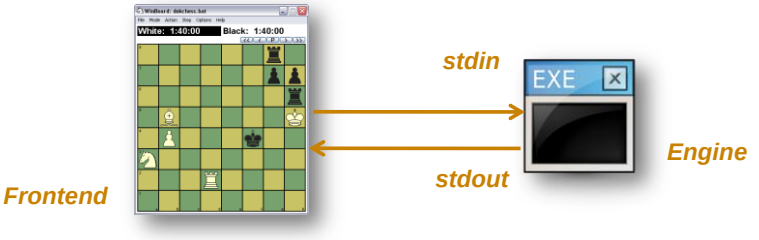
- Auf Windows lauffähig, idealerweise kostenfrei
- Anbindung einer eigenen Engine über ein geeignetes Protokoll möglich

Recherche ergibt:

- Mehrere Lösungen für verschiedene Betriebssysteme verfügbar
- sowohl frei als auch kommerziell
- 2 etablierte Protokolle

Protokolle für Schach-Engines/-Frontends

- 2 Lösungen etabliert/dokumentiert
- beide ASCII-basiert, beide via stdin/stdout



Universal Chess Interface (UCI)
http://wbec-ridderkerk.nl/html/UCIProtocol.html

Chess Engine Communication Protocol („Xboard/WinBoard“)
http://home.hccnet.nl/h.g.muller/engine-intf.html

Matt in drei Iterationen.
oose. Innovative Informatik

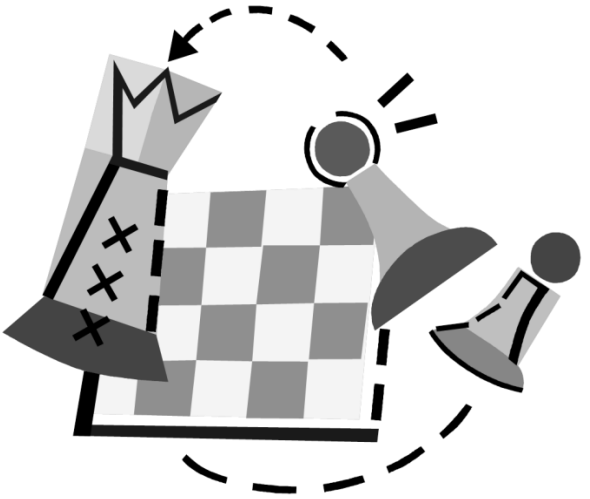
Betrachtete Schach-Frontends

Arena 3.0
http://www.playwitharena.com

Fritz for Fun 6
http://www.chessbase.de/

WinBoard 4.4.4
http://tim-mann.org/xboard.html

Demo: Eine Partie gegen DokChess::Iteration1



Agenda

3

2. Iteration („Bauerndiplom“), Steckbrief

Ziel:

Die Engine spielt Partien korrekt.

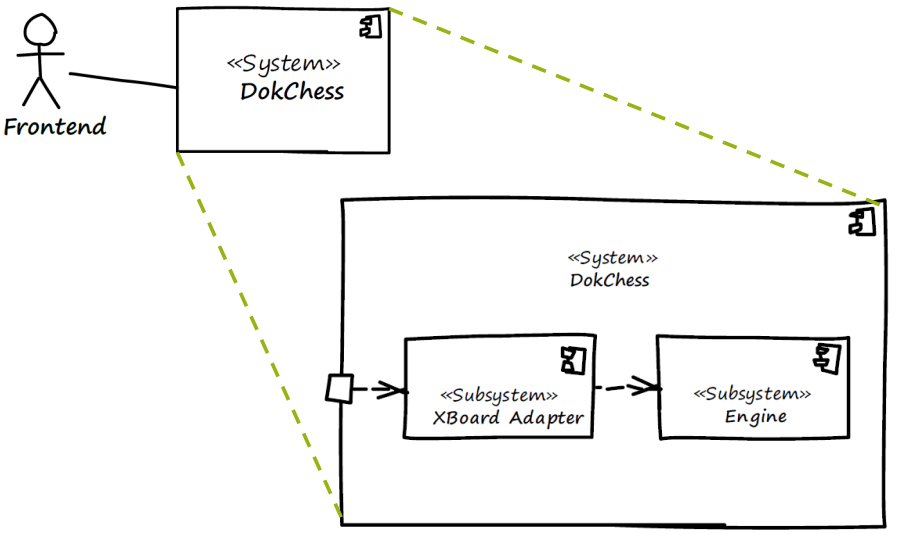
Zentrale Entscheidungen:

- Grundlegende Zerlegung in Subsysteme (Grundriss)
- Festlegung von Abhängigkeiten zwischen diesen

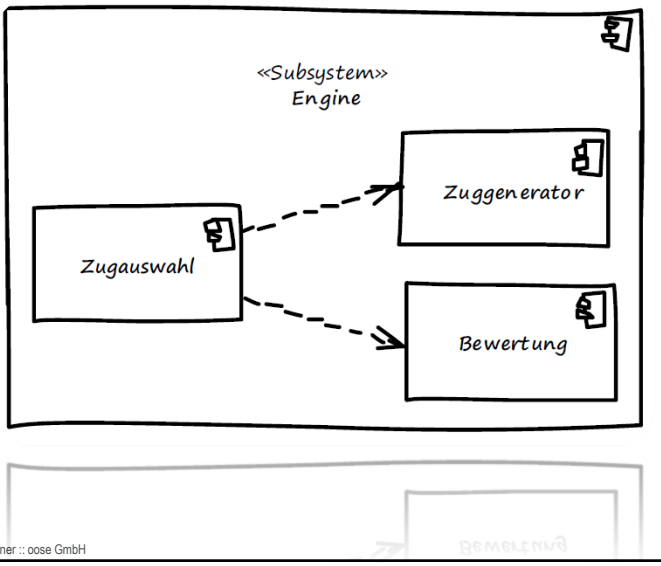
Implementierungsaufgaben

- Spielregeln

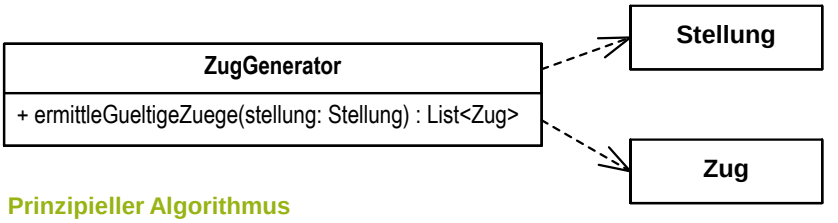
Bausteinsicht



Bausteinsicht Engine-Subsystem, Ebene 1



Zentrale Implementierungsaufgabe: Der Zuggenerator



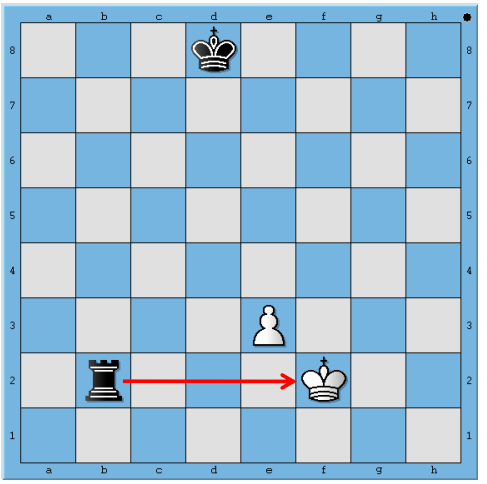
Prinzipieller Algorithmus

- Ermittle Farbe am Zug (aus Stellung)
- Prüfe für jede Figur der Farbe, wo sie überall hinziehen kann (freie Felder, ggf. schlagen einer gegnerischen Figur)
- Füge jeweils einen Zug in die Ergebnisliste

Implementierung

Im Grunde einfach, aber sehr aufwändig (bei mir: 10 Klassen 580 TLOC)
(6 verschiedene Figurenarten, Rochade, en passant)
Überprüfung auf gültige Züge schwierig wg. Schachgebot des Gegners

Problem Schachgebot (Einfaches Beispiel: Fesselung)



Weiß am Zug.

Lokal betrachtet:

e2-e3 und e2-e4 sehen gut aus (Zielfelder frei)

Nach Ausführung e2-e3:

Weiss im Schach

Züge sind beide ungültig!

Dürfen nicht in Ergebnisliste!

Einfache (naive?) Lösung

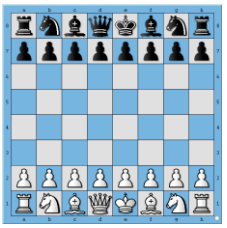
Prinzipieller Algorithmus

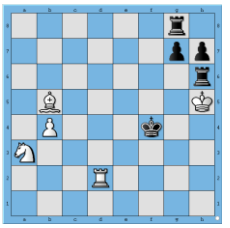
- Ignoriere Problem mit Schachgebot zunächst
  - ➔ Liste von Zugkandidaten („pseudolegaler“ Züge)
- Für jeden Zugkandidaten:
  - Führe Zug aus
  - Prüfe neue Stellung auf Schachgebot.
    - Ja: ➔ Zug kann verworfen werden (ungültig)
    - Nein ➔ Zug kann in Ergebnisliste

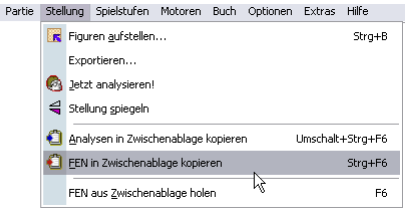
Praxistipp : Forsyth-Edwards-Notation

„ Die Forsyth-Edwards-Notation (FEN) ... ist eine Kurznotation, mit der jede beliebige Brettstellung im Schach niedergeschrieben werden kann.“
wikipedia.de

2 Beispiele:

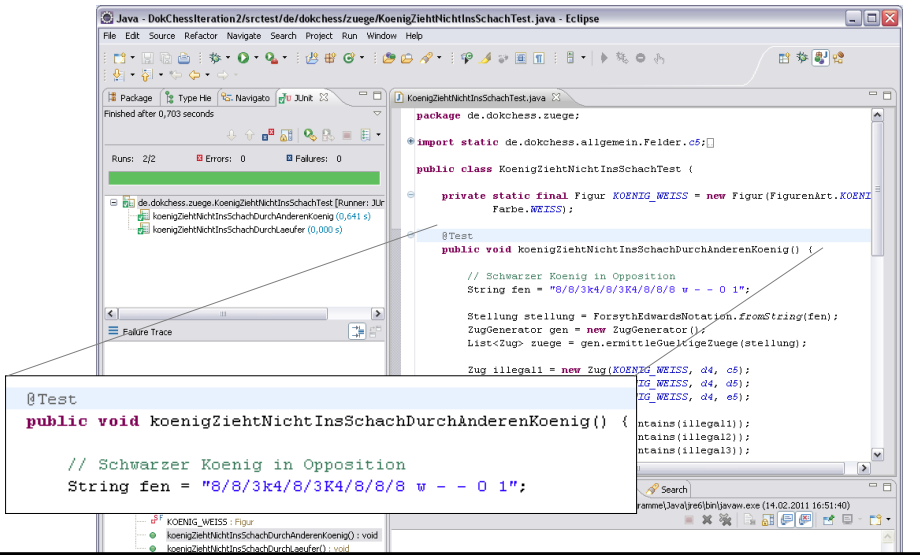




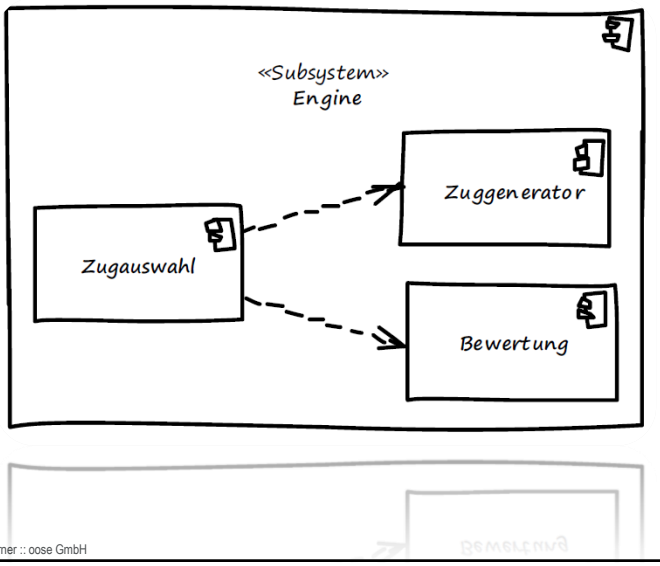


1. `"rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1"`
2. `"6r1/6pp/7r/1B5K/1P3k2/N7/3R4/8 w - - 30 79"`

Einsatz von FEN in Unit-Tests



Bausteinsicht Engine-Subsystem, Ebene 1



Naive Implementierungen für Bewertung und Auswahl

Bewertung einer Stellung anhand des Materials

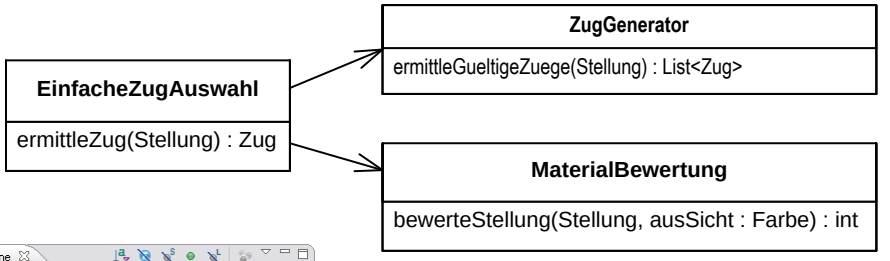
- Jede Figur erhält einen Wert (z.B. Bauer 1, … Dame 9)
- Eigene Figuren zählen positiv, gegnerische negativ
- Werte aufsummieren, je höher der Wert, je besser die Stellung

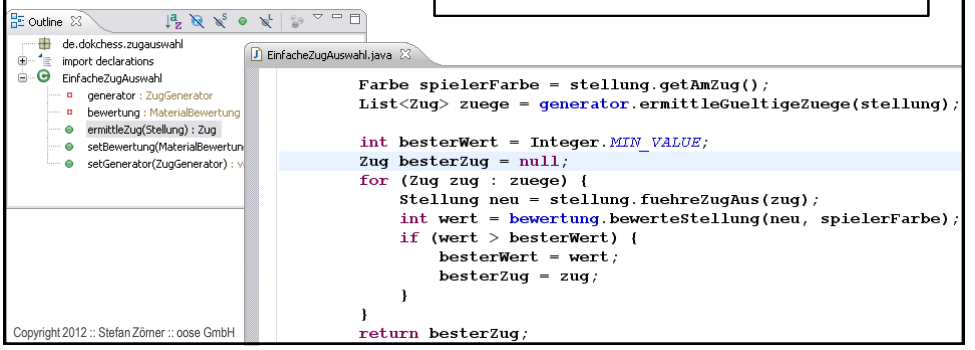
1 3 3 5 9 -

Einfache Auswahl aus der Liste der gültigen Züge:

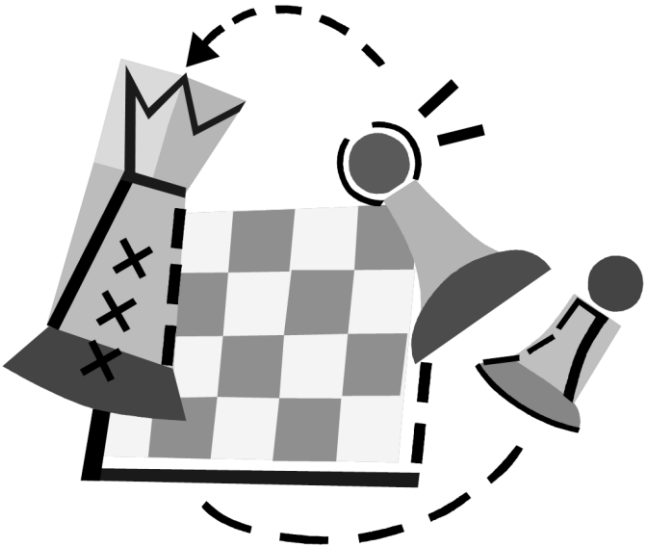
- Jeden Zug auf die aktuelle Stellung anwenden
- Resultierende Stellung bewerten (s.o.)
- Zug mit bestem Ergebnis wird ausgewählt

Klassendiagramm der Engine, Implementierung





Demo: Eine Partie gegen DokChess::Iteration2



Agenda

1. Die Aufgabe
2. Iteration 1: „Der Durchstich“
3. Iteration 2: „Das Bauerndiplom“
4. Iteration 3: „Der Taktikfuchs“
5. Ausblick und Weitere Informationen

4

Matt in drei Iterationen.
oose. Innovative Informatik

3. Iteration („Taktikfuchs“), Steckbrief

Ziel:

Die Engine spielt sinnvolle Partien.

Zentrale Entscheidungen:

- Auswahl der Algorithmen für Stellungsbewertung und Zugauswahl (Architekturentscheidungen?)

Implementierungsaufgaben

- Zugauswahl, Bewertung

Minimax-Algorithmus – Ein bisschen Theorie

"Der Minimax-Algorithmus ist ein Algorithmus zur Ermittlung der optimalen Spielstrategie für bestimmte Spiele, bei denen zwei gegnerische Spieler abwechselnd Züge ausführen. Die Minimax-Strategie sichert … höchstmöglichen Gewinn bei optimaler Spielweise des Gegners."

http://de.wikipedia.org/wiki/Minimax-Algorithmus

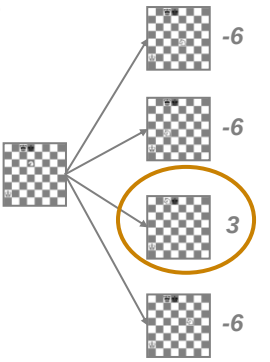
Die Natur des Schachspiels

- nicht vom Zufall abhängig
- offen, d. h. in jeder Spielsituation sind jedem der beiden Spieler alle Zugmöglichkeiten des jeweiligen Gegenspielers bekannt

Minimax – Bereits in Zugauswahl in Iteration 2

Schritte bisher

- Ermittle aus aktueller Stellung alle mögliche Züge
- Mache jeweils Zug, und bewerte neue Stellung
- Wähle das Maximum aus



Problem

- Reaktion des Gegners wird ignoriert
- Und „meine" Reaktion auf diese Reaktion, usw.

Lösungsidee

- Ermittle Suchbaum mit eigenen und gegnerischen Züge
- Bewerte Blätter (Baum bis zu bestimmter Tiefe)
- Finde jeweils den für mich/den Gegner besten Zug.

Minimax, 2 Halbzüge

Berechnung des Spielbaums

- fixe Tiefe, hier 2

Bewertung der „Terminalknoten“

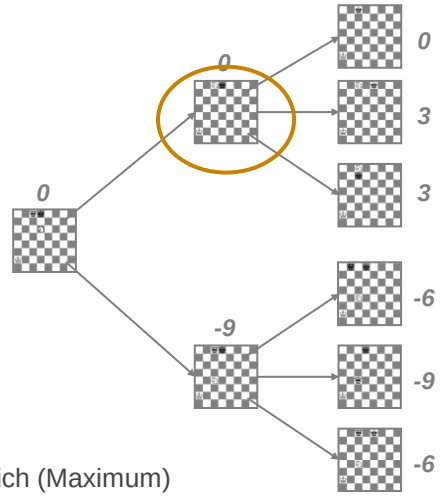
- Anwendung der Bewertungs-funktion aus „meiner“ Sicht

Minimum für den Gegner

- Was ist jeweils der beste Zug für den Gegner (Minimum)

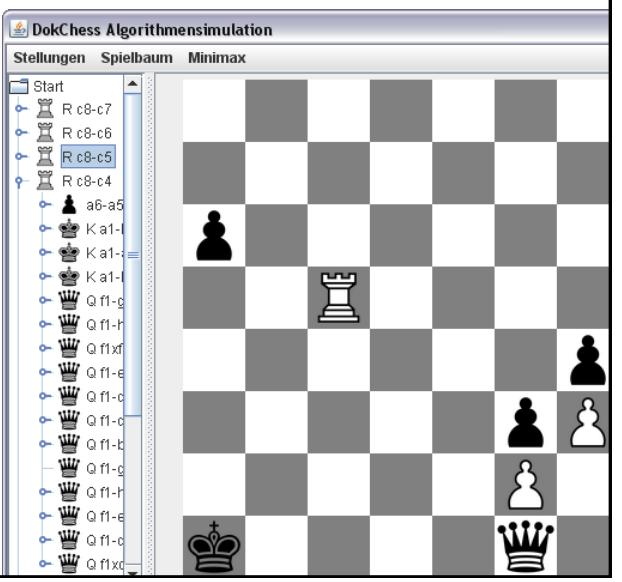
Maximum für mich

- Was ist jeweils der beste Zug für mich (Maximum)



Matt in drei Iterationen.

Demo Spielbaum + Minimax



Matt und Patt

Ein letztes, aber gravierendes Problem

- Minimax mit fester Tiefe und rein materialbasierte Bewertung kennt weder Matt noch Patt
- Konsequenz: Das Programm gewinnt Material, aber keine vermutlich Partie

§

„Das Ziel eines jeden Spielers ist es, den gegnerischen König so anzugreifen, dass der Gegner keinen regelgemäßen Zug zur Verfügung hat. Der Spieler, der dieses Ziel erreicht, hat den gegnerischen König **mattgesetzt** und das Spiel gewonnen."

„Die Partie ist remis (unentschieden), wenn der Spieler, der am Zuge ist, keinen regelgemäßen Zug zur Verfügung hat und sein König nicht im Schach steht. Eine solche Stellung heißt **Pattstellung**."

FIDE-Schachregeln

Matt in drei Iterationen.

Eine Lösung für Matt und Patt

Lösung im Minimax-Algorithmus

- Falls der Zuggenerator beim Explorieren keine gültigen Zuge für eine Stelle findet:

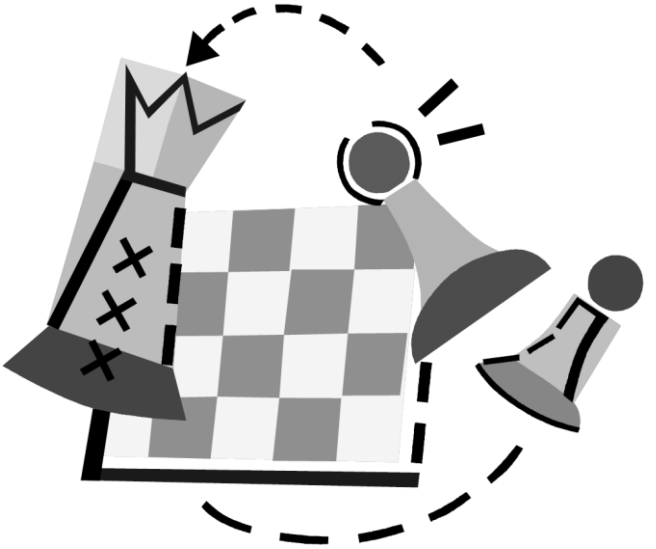
  Prüfe, ob die Seite am Zug im Schach steht

  Ja ➔ Matt, bewerte Knoten maximal bzw. minimal (je nach Sicht, wenn ich selbst Matt bin minimal)

  Nein ➔ Bewerte den Knoten ausgeglichen (Wert: 0)

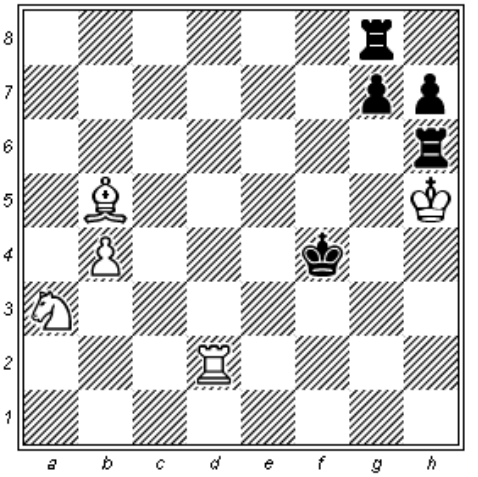
So wird ein Matt einem Materialgewinn vorgezogen, das eigene Matt wenn möglich verhindert, Patzer vermieden, …

Demo: Eine Partie gegen DokChess::Iteration3



DokChess gegen Fritz DS

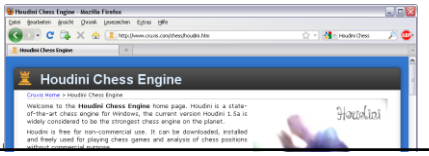
Fritz DS - DokChess Iter 3
Buchholz 2011



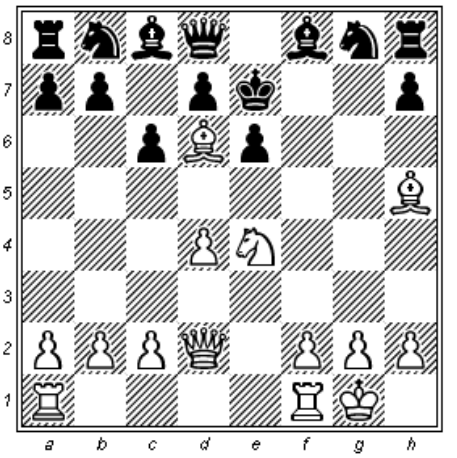
Schwarz setzt nach 78 Zügen matt.

DokChess gegen Houdini

http://www.cruxis.com/chess/houdini.htm



Houdini 1.5a - DokChess Iter 3
Buchholz 2011

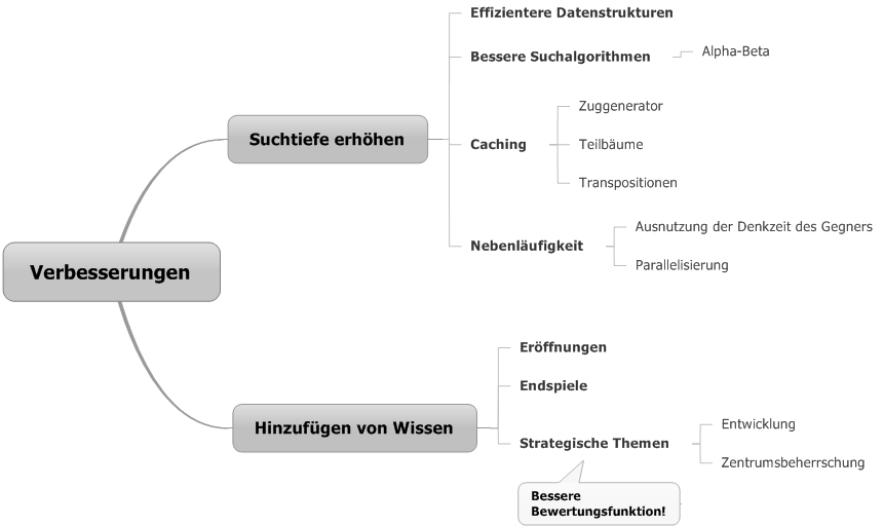


Weiß setzt nach 18 Zügen matt.

Agenda

5

Nächste Schritte



Fallbeispiel DokChess im Internet

- Architekturüberblick gegliedert nach arc42
- Quelltexte, Links, etc.

http://www.dokchess.de/

Das Buch zum Film ….

Softwarearchitekturen dokumentieren und kommunzieren.

Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten

von Stefan Zörner
Verlag: Hanser, Mai 2012
Sprache: Deutsch (ca. 280 Seiten)
ISBN-13: 978-3446429246

- Erfahren Sie, wie die Dokumentation der Architektur von der lästigen Pflicht zu einem integralen Kommunikations- und Arbeitsmittel wird.
- Lernen Sie architekturrelevante Einflussfaktoren und zentrale Entscheidungen festzuhalten.
- Erleben Sie am Beispiel einer Schach-Engine, wie eine nachvollziehbare Architektur entsteht.

Vielen Dank!

Ich freue mich auf Eure Fragen!

Stefan.Zoerner@oose.de

Lust auf neue Herausforderungen?

...dann hier entlang!

oose.
Innovative Informatik