Chess Online

Exercises in Domain Driven Design

# Scenario

A company named “Fide” has decided that an own platform to play chess online would perfectly fit within their corporate goals. Below, you can read what functionality is desired for such website.

## Functionality

Membership of our online Chess Club will be free. Visitors can register their membership on our site, preferably using an existing account (maybe a Google or Facebook account).

Only members can propose a new game. They can do so in two ways:

1. They can propose an *invitational game* by inviting a specific member (by email address or by member name) or by inviting a non-member (by email address).
2. They can propose an *open game* that can be joined by any other member. In the latter case, the proposing member can optionally set an Elo-rating range for the joining member, for instance 100 above and below his/her own rating.

If a game is joined, the system can start the game by randomly assigning a color to both players. Once the game has started only these two players can take turns. They take turns with a frequency ranging from once or twice a day to as often as every few seconds. In any case, time is recorded from the time they see the new board position to the time they submit their next valid move. The System should check that all moves are legal. Apart from making moves, players should also have the opportunity to propose a draw or resign. A player that submits no move for more than three days is also considered to have resigned the game.

When the game is over, the system should calculate the new Elo-rating for each player.

Chess rules:

<https://en.wikipedia.org/wiki/Chess>

An Elo-rating is the official rating of the International Chess Organization:

<https://en.wikipedia.org/wiki/Elo_rating_system>

# Event Storming

Work out an event storming session that visualizes the system that is to be built:

1. Add domain events [orange]
2. Add definitions and concerns [light green] and [purple]
3. Add Commands (with user roles, if appropriate) [blue]
4. Add External Systems [red]
5. Add Policies [Lilac]
6. Add Read Models [Green]
7. Add Aggregates [Yellow]

# Context Mapping

Create a context map

# Create Domain Model

Sketch 3 domain models for the services that governs the gameplay (the part includes that the system should check that all moves are legal).

Implement the most promising one in code

# Implement Event Sourcing

Implement event sourcing for the main aggregate

# Finish a skeleton Micro Service

Turn the application in a working service