Chess a Messaging Assignment: Architecture Document

Ryan van den Bogaard

2221121

i292181

Fontys University of Applied Sciences

Abstract

The following document is made for describing the Enterprise Architecture model.

Contents

Messaging Assignment: Architecture Document

Enterprise Architecture Model

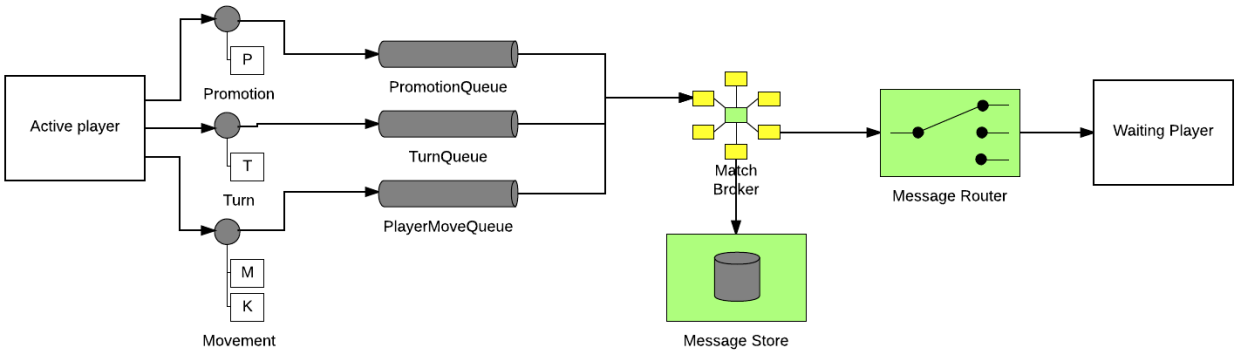

Figure 1. Model

The goal is to create and play a game using Unity3D with a C# compliant Messaging API (currently Apache NMS based on the OpenWire framework that works with Apache's ActiveMQ) and have a JMS compatible broker (ActiveMQ Artemis which supports JMS and OpenWire as well as a few other popular Messaging APIs).

In this model:

- Both the Active and Waiting Player are Unity3D/Game clients bound by one another through a Match Broker.
- Due to the basic rules of Chess, which allows the user to castle (which swaps the king position with a rooks' position and scooching the rook over to the left or right depending on the position on the board). I had to separate the 'Turn' and 'Movement' into separate messages for which I currently have split a single queue for easier handling. Also, when a chess piece hits the eight lane from its point of view you can promote the pawn into a Bishop, Knight, Rook and a Queen. Which is a separate thing on its own, thus I added a third channel for handling that.

- For future proofing and possible handling of the current standing of the game all messages will be stored. The plan was to implement a spectator mode and revision mode to check previously made steps.

Possible changes/updates to the model

Loading a game

Currently the game will load with any existing messages that were stored if connected. If there aren't any then the client using the White pawns will start. A change which might occur is the use of a new channel(s) or alteration of the Movement message for sending and collecting the currently available pawns on the playfield. This reduces the stress on recreating the game within Unity as well as possibly replacing a larger chunk of the message storage.