Reflexion, Plan vs. Wirklichkeit

Wahrend meiner Client Implementierung habe ich den Server ein bisschen besser verstanden, und habe mich dafür entschieden, dass ich den Server ein bisschen anders als zuvor geplant implementiere.
In der Vorwurfsphase habe ich mehrere Controllern (wie bei der Client) gezeichnet, aber mein Server hat jetzt nur einen Controller, der “GameStateController” genannt wurde.
Der Grund dafür ist, dass ich finde, dass ein Controller reicht fur den Zweck hier: der kontrolliert die Spiele, die Spielern und deren Zusammenhang.
Außerdem, habe ich den anderen Klassen fast gleich wie in dem Klassendiagramm gelassen.
Das Netzwerk-Converter Konzept war sehr leicht zu implementieren wegen der schon implementierte Client.
Das Validieren der HalfMaps habe ich meistens von dem Client genommen, was ein großes Vorteil war.
Dieses Mal habe ich drei Haupt Modelle: den GameData, den Player, und den InternalFullMap und einige sekundäre Modelle wie InternalHalfMap, MapNode, Coordinates, die bei der Haupt Modelle Konstruktion helfen.
Ich habe jedes Players’ HalfMap auch gespeichert, sodass das Validieren einfacher ist.
Die enums habe ich zusammen in einem Package gesetzt.

Die UML Klassendiagramm hat mir dazu geholfen, mir eine generelle Idee über den Klassen der Server zu machen, aber wegen unvollständige Verständnis über die Funktionsweise der Server, habe ich am Ende den Struktur geändert. Diese Änderungen waren viel schneller als beim Client.
Es war insgesamt nicht als mühsam wie die Client Implementierung und die Tests Generierung war viel einfacher.