TicTacToe_Gruppe2

Reflexion

David Bitterli:

Die Projektarbeit, welche sich um das Programmieren eines TicTacToe-Spiels handelte, war ziemlich anspruchsvoll. Wir hatten einen relativ guten Start mit den User Stories und Akzeptanzkriterien. Das Klassendiagramm und das Sequenzdiagramm stellten dann schon eine grössere Herausforderung dar. Das Programmieren des TicTacToe-Spiels an sich war dann aber ziemlich schwierig. Ich habe den Grossteil des Codes programmiert, jedoch habe ich den Code nicht direkt in das GitHub-Repository gepusht, sondern Abel geschickt. Es wäre sinnvoller gewesen, wenn wir mehr mit GitHub gearbeitet hätten.

Eine grosse Baustelle in unserem Programm war meiner Meinung nach die Undo-Funktion. Manchmal wurde sie nicht korrekt ausgeführt; ein anderes Mal wurde das Spielfeld falsch angezeigt usw. Schlussendlich konnten wir aber die Probleme beheben und die Undo-Funktion korrekt implementieren. Zudem habe ich ab und zu Methoden programmiert, die ich im Anschluss gar nicht mehr brauchte und die dadurch eher als Code-Smell auftraten.

Ich würde behaupten, dass trotz einiger Fehltage von Abel die Arbeit relativ gerecht aufgeteilt wurde und wir uns gut ergänzt haben. Für die Zukunft weiss ich, dass es besser ist, ein solches Projekt früher anzugehen und nicht nach hinten zu verschieben.

Abel Solomon:

Mein Schwerpunkt lag auf den Klassendiagrammen, Sequenzendiagrammen und dem Testen. Da ich während der Theoriephase zu MVC gefehlt habe, war es anfangs schwierig, die Trennung zwischen Model, View und Controller richtig umzusetzen. Besonders bei der Erstellung der Diagramme mussten wir einige Anpassungen vornehmen, um die Verantwortlichkeiten klarer zu definieren.

Beim Testen habe ich Unit- und Integrationstests geschrieben, um die Spiellogik abzusichern. Dabei lag der Fokus auf den Gewinnbedingungen, dem Spielerwechsel und der Validierung von Zügen. Durch die Tests konnten wir einige Fehler frühzeitig erkennen und beheben.

Die Zusammenarbeit hätte effizienter sein können, wenn wir GitHub konsequenter genutzt hätten, anstatt Code direkt auszutauschen. Insgesamt habe ich viel über Softwarearchitektur und

die Bedeutung sauberer Strukturen gelernt.