Wann es sich lohnt ein Programm zweimal zu schreiben

Andreas Stadelmeier

28. April 2017

Das Softwarprojekt "JavaTX"war in einer Spirale immer schlechter werdenden Quellcodes gefangen. Die Liste der Fehler im Bugtracker nahm stetig zu. Das Debuggen, also Finden von Ursachen für Fehler, wurde immer aufwendiger. Auch die Bugfixes an sich wurden immer komplexer und führten meist zu neuen Fehlern. Allein das Finden und Beheben von Bugs wurde dadurch zur Hauptaufgabe, obwohl noch genügend Änderungen und Erweiterungen ausstanden oder sich in Planung befanden.

Als also klar wurde, dass durch einfaches Refactoring diesen Problemen nicht mehr beizukomen war, entschieden wir uns das Programm nochmal von Grund auf neu zu schreiben.

Mein Beitrag gliedert sich wie folgt:
Zuerst wird das Projekt und seine Entstehungsgeschichte vorgestellt. Anschließend beantworte ich zuerst die Frage, wie das Projekt in den oben beschriebenen Zustand geraten konnte. Welche Designfehler sich durch das Projekt gezogen haben. Wieso Debugging teilweise unmöglich wurde.

Danach versuche ich auch die Gründe zu analysieren. Wie sind diese Fehler entstanden? Wieso konnten sie sich so lange im Projekt halten? Dies erläutere ich auch mithilfe von konkreten Fallbeispielen aus dem Projekt. So zeige ich zum Beispiel wie es dazu kommen konnte, dass bei jedem Programmdurchlauf über 4000 Zeilen Logausgaben produziert wurden.

Zuletzt zeige ich mögliche Auswege aus der Krise und präsentiere die Variante, für die wir uns schlussendlich entschieden haben. Zusätzlich stelle ich auch die aus diesen Fehlern gewonnenen Erfahrungen vor. Welche Auswirkungen es haben kann, wenn Grundsätze aus der objektorientierten Softwareentwicklung nicht eingehalten werden. Wie einfach es ist in einen solchen Zustand zu gelangen und wie schwierig der Ausweg daraus sein kann.