Einleitung

Grady Booch gibt in seinem Paper „The History of Software Engineering“ eine gute Übersicht über die Entwicklung des Begriffs „Software Engineering“. Im folgenden Absatz ist das Paper zusammengefasst. Laut Booch wurde der Begriff erstmals 1963/64 von Magaret Hamilton erwähnt. Als damalige Lead-Developerin in Skylab und Apollo wollte sie ihre Arbeit von „Hardware Engineers“ unterscheiden. In den 1960er soll es drei große Faktoren gegeben haben, die die Softwareentwicklung beeinflussten, und zwar die Kommerzialisierung von Software als Produkt, die Entwicklung des SAGE-Verteidigungssystems und das US-Raumprogramm und die daraus resultierenden überlebenswichtigen Softwaressyteme. Zwischen den 60er und 80er wurden die Konzepte wie Modular Programming, Wasserfall Modell und Entity-Relationship-Modell für die Softwareentwicklung entwickelt. In den 80er und 90er veränderten sich die Ansprüche auf Softwaresysteme durch die zunehmende Globalisierung. Daher mussten neue Konzepte wie UML, Spiral Modell, OOP eingeführt werden, um die wachsenden Anforderungen zu bewerkstelligen. Durch das Internet in den 90er erhielt die Rolle der Software abermals eine neue Stellung. Der „Consumer“ hatte nun als neuer Stakeholder andere Ansprüche an die Software als Unternehmen. Zu dieser Zeit haben sich bereits Techniken der Softwareentwicklung etabliert, wie „Continuous-Integration“ und iteratives Entwickeln. Hinzu kamen nun auch Design Pattern auf Architektur- und Code-Ebene. Das *Internet-of-Things* gewann durch Smartphones an Bedeutung, sodass monolitische Software durch Microservices, Webtechnologien und Cloud Computing ersetzt wurden. (Booch, 2018)

Wie zu sehen ist, hat das Wort „Software Engineering“ eine lange Entwicklungsgeschichte. Sie zeigt, dass durch die wachsenden Anforderungen und die wachsende Komplexität, Software gut organisiert, gemanagt und durch geeignete Richtlinien, wie Patterns, Agiles Programmieren und gute Dokumentation entwickelt werden muss.

Die moderne Softwareentwicklung baut auf fünf Phasen auf. Die Analyse, das Design, die Codierung, das Testen, und die Wartung. (Sommerville, 2011) Jeder dieser fünf Phasen nutzt Techniken und Strategien, die in den letzten 60 Jahren entwickelt wurden. In der Analyse gibt es Stakeholderlisten und Anforderungsanalyse, aus der sich funktionale und nicht funktionale Anforderungen ableiten lassen. Aus diesen lassen sich das Design entwickeln, in dem man aus den Anforderungen UML-Diagramme und Softwarearchitekturen entwickelt. Hinzu kommen Grundsätze und Leitsätze, die die Gestaltung von Bedienoberflächen und Dialogen festlegen.

Dadurch lassen sich den Stakeholderlisten nahe Software entwickeln. In der Implementierungsphase haben sich bestimmte Frameworks, und Code-Richtlinien, wie YAGNI, DRY, KISS, SOLID durchgesetzt, um eine hohe Qualität des Codes zu gewährleisten. (DRY vs KISS – Clean Code Prinzipien, 2018; t2informatik GmbH, 2022; Oloruntoba, 2021)

Genauso ist das Testen und Warten von Software sehr wichtig. Software spielt in vielen Lebenslagen der Menschen eine wichtige Rolle und würde bei einem Ausfall zu fatalen Folgen führen. Um die Software zu verbessern, werden daher auch oft *Refactoring* betrieben. Für die Realisierung von Software werden dann auch sowas wie Agiles Programmieren, Scrum und Kanban verwendet.

Das Ziel dieses Projektes ist es zu zeigen, dass durch eine gute Planung und unter Einhaltung moderner Programmier-Richtlinien eine releasefähige Software im Rahmen des Studienprojektes in einem angemessenen Zeitraum erstellt werden kann. Dabei wurde für die Entwicklung eines Spiels namens „Harsh-Politics“ entschieden. Das Spiel baut auf altbekannte Spiele auf, wie Tekken und andere diverse Kampfspiele, in der zwei Spieler gegeneinander kämpfen und unter anderem verschiedene Fähigkeiten besitzen. In diesem Projekt sollen bekannte Politiker gegeneinander kämpfen. Dabei besitzt jeder Politiker typische Wiedererkennungsmerkmale, die unter anderem auch als Spezialfähigkeit eingesetzt werden kann. Mit diesem Spiel werden die Phasen der Softwareentwicklung in den folgenden Kapiteln beispielhaft verdeutlicht.