

## Einleitung / Motivation / Einführung / Hintergründe

Der Begriff „Software Engineering“ wurde erstmals 1963/64 von Margaret erwähnt. Als damalige Lead-Developerin in Skylab und Apollo wollte sie ihre Arbeit von „Hardware Engineers“ unterscheiden. In den 1960er gab es drei große Faktoren, 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 Softwaresysteme. 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. Es mussten neue Konzepte wie UML, Spiral Modell, OOP eingeführt werden, um den wachsenden Anforderungen Herr zu werden. 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 CI und iteratives Entwickeln. Hinzu kamen nun auch Design Pattern auf Architektur- und Code-Ebene. Das Internet-of-Things gewann durch Smartphones an Bedeutung, sodass monolithische Software durch Microservices, Webtechnologien und Cloud Computing ersetzt wurden. []

„Software Engineering“ hat eine lange Geschichte. Sie zeigt, dass durch die wachsenden Anforderungen und die wachsende Komplexität, Software gut organisiert, gemanaget und durch geeignete Richtlinien, wie Patterns, Agile Programming, Documentation werden muss. Moderne Softwareentwicklung baut auf 4 Phasen. Die Analyse, das Design, die Codierung, das Testen, und die Wartung. Jeder dieser 4 Phasen nutzt Techniken und Strategien, die in den letzten 20 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 die Frameworks, und Code-Richtlinien, wie YAGNI, DRY, KISS, SOLID durchgesetzt, um eine hohe Qualität des Codes zu gewährleisten. 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. Zeit spielt in der Softwareentwicklung auch eine bedeutendere Rolle. Zeitmanagement wichtig. Für die Realisierung von Software werden dann auch sowas wie Agile Programming, 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 „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.