# TITEL DES EXPOSÉS

**Exposé zur Seminararbeit**

Johannes Teklote
jotek001@stud.fh-dortmund.de

7091992

22. November 2020

## 1 Motivation

Laut [Bal09] entfallen 80% des Aufwands in der Softwareentwicklung auf deren Wartung, wobei 40% davon benötigt werden, um die zu wartende Software zu verstehen. Ein erheblicher Teil der Kosten für die Softwareentwicklung entfällt damit auf das Verstehen der Software. Wenn insgesamt die Entwicklungskosten reduziert werden sollen, stellt dieser Umstand einen wesentlichen Hebel dar. Denn der Wartungsaufwand im Übrigen ist vielfach kaum zu verringern, weil sich die Anforderungen an die Software im Laufe der Zeit ändern, weil auftretende Probleme behoben werden müssen oder weil Abhängigkeiten auf andere Software, Frameworks oder Komponenten angepasst werden müssen. Bereits durch das Erfüllen gewisser Qualitätsstandards lässt sich jedoch der für das Verstehen des Codes notwendige Aufwand reduzieren.

Qualitativ hochwertige Software zeichnet sich nach [ISO11] dadurch aus, dass sie stabil ist und gut gewartet, analysiert, erlernt und verändert werden kann. Software, die diese Anforderungen erfüllt, beinhaltet Code, der nach bestimmten Richtlinien gebildet wird. Zu diesen Regeln gehört beispielsweise, dass Code auf einer Ebene einheitlich eingerückt wird, dass öffnende geschweifte Klammern in der gleichen Zeile stehen wie der Code, der die Klammer erfordert, dass Variablennamen sprechend sind und nicht nur aus einem Buchstaben bestehen oder dass Parameter auf ihre Gültigkeit überprüft werden, bevor mit ihnen gearbeitet wird.

Die Einhaltung solcher Regeln scheint auf den ersten Blick nur von geringer Bedeutung zu sein, allerdings lenken beispielsweise Unstimmigkeiten in der Formatierung beim Lesen des Codes ab und sie erschweren es, sich auf den eigentlichen Code zu konzentrieren. In [Pra15] und [Spi11] werden diese Formatverstöße als Hintergrundrauschen beschrieben, das vom eigentlichen Code ablenkt.

## 2 Problemstellung

Code, der nach den oben beschriebenen Regeln geschrieben wurde, ist laut [Pra15] leichter verständlich und dadurch auch leichter erlernbar und veränderbar. Um die Teammitglieder dazu zu bringen, sich an die vereinbarten Regeln zu halten, gibt es zwei Möglichkeiten. Zum einen kann im Reviewprozess des Projekts ein automatischer Test eingebaut werden, der dafür sorgt, dass grundsätzlich funktionierender Code dann nicht angenommen wird, wenn er gegen festgelegte Regeln verstößt, wenn er also beispielsweise Stylingrichtlinien nicht einhält oder es potentiell unsichere Variablenzugriffe gibt. Zum anderen können die Teammitglieder durch Belohnungen dazu motiviert werden, von sich aus die festgelegten Regeln einzuhalten und dies auch selbst zu überprüfen. Es stellt sich nun die Frage, ob eine solche Motivation in durch Gamification erreicht werden kann. Bei Gamification werden Methoden genutzt, die aus Computerspielen bekannt sind. Dazu gehören beispielsweise Levels, Badges oder Leader Boards, die das Team oder ein einzelnes Teammitglied erreichen kann.

Das Ablehnen von eingereichtem Code auf Grund von Verstößen gegen Richtlinien führt dazu, dass die Teammitglieder die vereinbarten Programmierrichtlinien mit der Zeit von sich aus einhalten, ohne immer darauf hingewiesen werden zu müssen. Es ergibt sich also ein Lerneffekt, der bewirkt, dass die Teilnehmer auch in Zukunft qualitativ hochwertigen Code einreichen. Im Rahmen dieser Arbeit soll untersucht werden, ob beziehungsweise in wie weit auszuwählende Gamification-Elemente ebenfalls die Motivation erhöhen, qualitativ hochwertigeren Code einzureichen. Neben qualitativ hochwertigerem Code könnte so durch einen gewissen SSpaßfaktoräuch eine höhere Zufriedenheit der Teammitglieder und somit eventuell auch eine intrinsische Motivation zum Schreiben von qualitativ hochwertigerem Code erreicht werden.

## 3 Zielsetzung

Ziel dieser Arbeit ist die Erweiterung der Codeanalyseplattform coderadar um Gamification-Elemente zur Steigerung der Code-Qualität. Dazu werden zunächst Gamification-Elemente ausgewählt und in coderadar implementiert. Anschließend wird eine Balancierungsmatrix für eine gleichmäßige Verteilung der Punkte, auf denen Levels, Leader Boards und ähnliches basieren, erarbeitet. Abschließend wird ein Feldversuch durchgeführt, um festzustellen, ob und in wie weit die vorgestellten Maßnahmen die Softwarequalität nachhaltig verbessern konnten.

## 4 Vorgehensweise

Zunächst werden die Grundlagen dieser Arbeit vorgestellt. Es wird erläutert, was Gamification ist und welche Gamification-Elemente in dieser Arbeit Verwendung finden wer-

den. Daneben wird im Rahmen der Vorstellung verwandter Arbeiten darauf eingegangen, welche Implementierungen und Implementierungsansätze es bereits für Gamification-Elemente in Zusammenhang mit der Code-Qualität gibt. Außerdem wird die Codeanalyseplattform coderadar vorgestellt, die als Basis für die im Rahmen dieser Arbeit erfolgende Implementierung dienen wird.

Im Anschluss daran wird die Einbindung von Gamification-Elementen erläutert. Dazu gehört zunächst die Auswahl der Metriken, anhand derer die Codequalität bewertet werden soll. Entsprechend diesen Metriken wird der konkrete Code dann untersucht. Anhand des Ergebnisses dieser Untersuchung werden anschließend die Punkte verteilt, auf denen später Levels oder Leader Boards beruhen. Die Verteilung dieser Punkte erfolgt mit Hilfe einer Balancierungsmatrix, die für eine gleichmäßig gewichtete Punkteverteilung für unterschiedliche Aktionen sorgt. Diese Balancierungsmatrix wird ebenfalls an dieser Stelle definiert. Des Weiteren werden die Spielmechanismen, die Spielregeln und das Spielziel beschrieben.

Danach wird auf die Implementierung eingegangen. Zunächst werden die Anforderungen an die Implementierung erhoben und eine Definition of Done, also die Definition, wann das Projekt fertig ist, erstellt. Auf Basis dieser Anforderungen werden dann die erforderlichen Änderungen an der coderadar-Plattform modelliert. Es folgt die eigentliche Implementierung.

Anschließend wird die Evaluierung durchgeführt. Dazu wird zunächst ein Projekt zur Untersuchung in coderadar eingecheckt, und anhand der ausgewählten Metriken wird dessen Code-Qualität bewertet. Der hier ermittelte Wert soll als Grundlage für die Messung der künftigen Bewertungsänderungen dienen. Über eine gewisse Zeit wird das Projekt dann im Rahmen eines Feldversuchs verfolgt und es wird anhand der ausgewählten Metriken ermittelt, in wie weit sich die Bewertung des untersuchten Projektes verändert. Zusätzlich dazu sollen die Projektteammitglieder zu ihrer Arbeit mit den Gamification-Elementen in coderadar befragt werden.

Abschließend wird das Ergebnis der Arbeit zusammengefasst. Als Grundlage für dieses Fazit dienen dabei zum einen die Beobachtungen über die Veränderung der Code-Qualität und zum anderen die Ergebnisse aus der Befragung der Projektteammitglieder. Außerdem wird reflektiert, welche Herausforderungen es bei der Implementierung und der Durchführung des Feldversuchs gab. Zum Schluss wird ein Ausblick auf die weitere Arbeit gegeben.

## Literatur


[Bal09] Balzert, Helmut: *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*. Heidelberg : Spektrum Akademischer Verlag, 2009. http://dx.doi.org/10.1007/978-3-8274-2247-7_3. http://dx.doi.org/10.1007/978-3-8274-2247-7_3. – ISBN 978–3–8274–2247–7

[ISO11] International Organization for Standardization: System und Software-Engineering – Qualitätskriterien und Bewertung von System und Softwareprodukten (SQuaRE) – Qualitätsmodell und Leitlinien. 2011. – Standard

[Pra15] Prause, Matthias Christian und J. Christian und Jarke: Gamification for enforcing coding conventions, 2015, S. 649–660

[Spi11] Spinellis, Diomidis: elytS edoC. In: *IEEE Software* 28 (2011), März/April, Nr. 2, S. 104–103