Who is Who in Quality Engineering

Michael Thurner

27. Mai 2015

1 Quality Engineering - ein kurzer Überblick

Das Quality Engineering ist ein Gebiet der Informatik, dass sich dem hohem Qualitätsanspruch in den Bereichen des Managements, der Erstellung, des Betriebs und der Weiterentwicklung von IT-Systemen widmet[1].

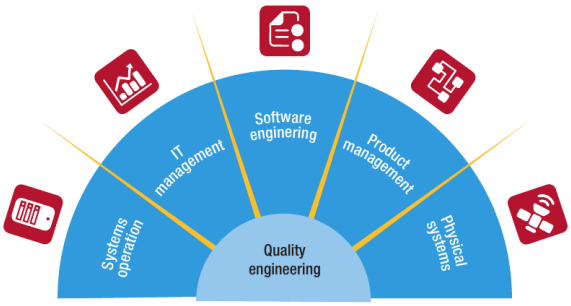

Abbildung 1: Bereiche des Quality Engineerings.

Durch die zunehmende Vernetzung von IT-Diensten über die Grenzen von Plattformen hinaus, ist es notwendig geworden die Qualität durch den Einsatz des Quality Engineerings zu gewährleisten. Die Gewährleistung erfolgt dabei unter Einbezug von Methoden aus den Bereichen des Software-Produktmanagements, des Software-Qualitätsmanagements, des IT-Security-Managements, des IT-Servicemanagements, des Software Engineerings und des System Engineerings. Diese Bereiche stehen dabei in Verbindung und sichern reziprok die Qualität.

Im Folgenden werden nun einige Personen, deren Beitrag die Entwicklung des Quality Engineerings vorantrieb, vorgestellt. Es handelt sich dabei nicht um eine Biografie, sondern um eine kurze Vorstellung ihrer bedeutendsten bzw. bekanntesten Werke.

2 Who is Who

In diesem Abschnitt wird über das Werk einiger Personen im Bereich des Quality Engineerings berichtet. Der Fokus liegt dabei einmal auf dem Software Engineering, der Programmverifikation und der Modellierung. Hierbei soll nur ein kurzer Überblick auf das Werk entstehen, ohne dabei zu sehr in die Thematik einzugehen.

2.1 Kent Beck

Kent Beck (* 1961) gilt als Begründer des Extreme Programmings[4]. In Zusammenarbeit mit Erich Gamma war er zuständig für die Portierung des Smalltalk-Testframeworks SUnit nach Java, dort unter dem Namen JUnit[5]. SUnit wurde im Übrigen von Beck selbst entwickelt.

Bei einem Unit-Test findet dabei eine einfache Überprüfung auf Werte und Bedingungen statt, sollten keinerlei Probleme auftreten, ist der Test erfolgreich. Sollte der Test fehlschlagen, wird der Programmcode solange abgeändert, bis das gewünschte Ergebnis eintritt.

```
public class Test {
  public String concat(String a, String b) {
     return a + b;
  }
}

import org.junit.Test;
import static org.junit.Assert;

public class UnitTest {

  @Test
  public void testConcat() {
     Test test = new Test();
     String concatenated = Test.concat("ab","cd");

     assertEquals("abcd",concatenated);
  }
}
```

Listing 1: Einfaches Beispiel eines JUnit-Test.

Listing 1 illustriert, wie ein Unit-Test aussieht. In diesem Fall wird die Richtigkeit einer Stringkonkatenation überprüft.

2.2 Tony Hoare

Tony Hoare (* 1934) erlangte Bekanntheit durch die Entwicklung des Quicksort-Sortieralgorithmus und des Hoare-Kalküls[3]. Mit Hilfe des Hoare-Kalküls lässt sich die Korrektheit von Algorithmen bzw. von Programmen nachweisen. Dies wird im folgenden kurz erklärt.

Nun folgen einige kurze Definitionen, die das Vorgehen erläutern. Im Zentrum steht das *Hoare-Tripel*[2] $\{Q\}\ P\ \{R\}$, dabei sei P ein while Programm, Q und R sind Zusicherungen, Q wird dabei als Vorbedingung und R als Nachbedingung bezeichnet. Um nun zu zeigen, ob ein Programm korrekt ist, muss mittels der in Tabelle 1 dargestellten Regeln, gezeigt werden, dass das Hoare-Tripel abgeleitet werden kann. Auf die genaue Vorgehensweise wird hierbei jedoch nicht eingegangen.

[z] $\dfrac{}{\{Q\{x \mapsto t\}\}\ x := t\ \{Q\}}$ [a] $\dfrac{\{Q'\}\ P\ \{R'\}}{\{Q\}\ P\ \{R\}}\ Q \models Q', R' \models R$

[s] $\dfrac{\{Q\}\ P_1\ \{R\} \quad \{R\}\ P_2\ \{S\}}{\{Q\}\ P_1; P_2\ \{S\}}$ [w] $\dfrac{\{I \wedge B\}\ P\ \{I\}}{\{I\}\ \mathit{while}\ B\ \mathit{do}\ P\ \mathit{end}\ \{I \wedge \neg B\}}$

Tabelle 1: Die Regeln des Hoare Kalküls.

2.3 Grady Booch, Ivar Jacobson, James Rumbaugh

Durch das gegenseitige Einbeziehen der Methoden von Grady Booch, Ivar Jacobson und James Rumbaugh entstand in den 1990er Jahren die *Unified Modeling Language (UML)*[6], die heute als führend im Bereich der Softwaresystem-Modellierung gilt.

UML dient dabei:

- zur besseren Kommunikation mit Mitarbeitern bzw. Kunden
- zur Darstellung komplexer Inhalte
- zur besseren Verständlichkeit

UML findet sich in Verwendung, für z.B. Klassendiagramme, ein kurzer Überblick, was dargestellt werden kann:

- Klassen
- Abschnitte von Klassen, wie z.B. Attribute, Methoden etc.
- die Beziehung zwischen den Klassen

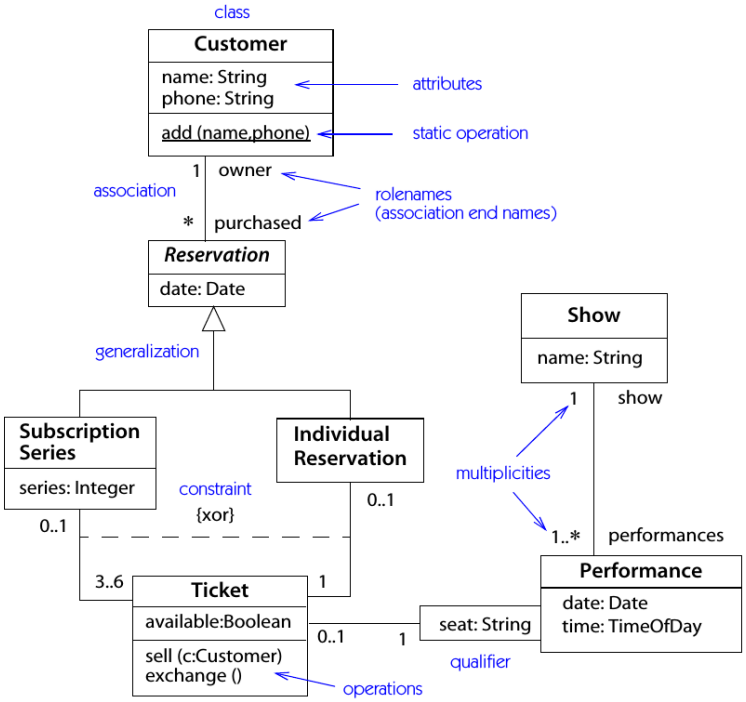

Abbildung 2: Beispiel eines UML Klassendiagramms.

3 Konklusion

Im Gebiet des Quality Engineerings findet sich eine Vielzahl an verschiedenen Akteuren, die alle ihren Beitrag leisten, um den Fortschritt auf diesem Gebiet voranzutreiben, hier wurde nur ein kurzer Überblick gewährt, der zeigen soll, welche Bereiche dabei eine Rolle spielen. Es wurden aber nicht alle Bereiche angeführt.

Literatur

[1] R. Breu, A. Kuntzmann-Combelles, M. Felderer: *New Perspectives on Software Quality.*, IEEE Computer Society, P. 32-38. January-February 2014

[2] G. Moser: *Einführung in die theoretische Informatik: Skriptum zur Vorlesung, 3. Auflage.*, UIBK, 2013

[3] T. Hoare: *An axiomatic basis for computer programming.*, Communications of the ACM 12, P. 576-580. October 1969

[4] K. Beck: *Extreme Programming. Das Manifest.*, Addison-Wesley, 2000

[5] K. Beck: *JUnit Pocket Guide*, O'Reilly, 2004

[6] G. Booch, I. Jacobson, J. Rumbaugh: *The Unified Modeling Language Reference Manual, Second Edition.*, Addison-Wesley, 2005