Ludwig-Maximilians-Universität München

Hausarbeit
- Juristisches IT-Projektmanagement -
Dozent: Dr. Frank Sarre

Quantifizierung nicht-funktionaler Anforderungen

vorgelegt von

Zhenhao Li

Wintersemester 2016/17

28.01.2017

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

In IT-Projekten werden nicht-funktionale Anforderungen wie Skalierbarkeit oder Wartbarkeit von Softwaresystemen oft nicht explizit erwähnt. Dies liegt vor allem darin, dass der Auftraggeber meistens aus einem anderen Fachgebiet stammt und sich somit nicht mit den informationstechnologischen Details auskennt. Dadurch wird der Fokus auf die gewünschten Funktionalitäten des Auftraggebers gelegt, welche von den funktionalen Anforderungen beschrieben werden. [Wie2003]

Jedoch spielen auch die nicht-funktionalen Anforderungen eine wichtige Rolle. Denn wenn beispielsweise eine erstellte Software nicht performant genug läuft, könnte das Geschäft des Auftragsgebers negativ beeinflusst werden. Schließlich kann es aufgrund nicht explizit definierter nicht-funktionalen Anforderungen zu Uneinigkeiten und Streitfälle kommen, die für beide Vertragsparteien teuer enden können.

1.2 Zielsetzung

Um präventiv gegen das Problem der nicht bzw. nicht konkret genug definierten nicht-funktionalen Anforderungen vorzugehen, zeigt diese Arbeit mögliche Lösungsansätze mittels Methoden aus dem juristischen IT-Projektmanagement. Dadurch können die nicht-funktionalen Anforderungen quantifiziert werden und schließlich konkret in den Verträgen festgehalten werden.

2 Grundlagen

2.1 Quantifizierung

Quantifizierung bedeutet unter Anwendung bestimmter Maßnahmen gewisse Eigenschaften eines Objekts messbar zu machen. [eWD2017] Beispielsweise ist die Aussage „das Auto ist rot“ nur eine qualitative Aussage, da man nicht messen kann, wie „rot“ dieses Auto ist. Um eine Farbe zu quantifizieren, gibt es als Maßnahmen beispielsweise verschiedene Farbmodelle. So kann dieses Rot des Autos gemäß dem RGB Farbmodell einen Wert von RGB (255, 20, 30) besitzen. Im Gegensatz zu der qualitativen Aussage kann man in diesem Fall die Farbe eindeutig zuordnen und auch reproduzieren.

Gerade im juristischen Kontext spielt die Reproduzierbarkeit eine wichtige Rolle. Denn in einem Streitfall bei Softmängeln muss der Beklagter darlegen können, in wieweit etwas nicht funktioniert. Auch der Angeklagter will zeigen können, dass die Umsetzung vertragsgemäß erfolgt ist.

Die Quantifizierung der nicht-funktionalen Anforderungen von Softwaresystemen ist sowohl für den Auftraggeber als auch für den Auftragnehmer interessant. Denn der Beweislast liegt je nach Vertragsart entweder beim Auftraggeber oder beim Auftragnehmer. In Fall eines Werkvertrags liegt der Beweislast zunächst beim Auftragnehmer, dieser wechselt nach Abnahme der Software zum Auftraggeber. [Fra2017] Der Auftraggeber kann anhand der Quantifizierung im Vertrag die Anforderungen konkreter festlegen, so dass die vereinbarten Leistungen bei der Abnahme auf Vertragsmäßigkeit überprüft werden können. Der Auftragnehmer kann bei klar definierten Werten sein Projekt besser planen und vermeidet bei einer vertragsgemäßen Umsetzung Streitfälle.

2.2 Nicht-funktionale Anforderungen

Während die funktionalen Anforderungen die zu leistenden Funktionalitäten der Software angeben, werden bei den nichtfunktionalen Anforderungen vor allem verschiedene Qualitätsmerkmale beschrieben. [Suz2012]

Eine eindeutige Auflistung dieser Qualitätsanforderungen existiert nicht. Dennoch wird ein standardisiertes Qualitätsmodell angestrebt, das dann zur Bestimmung nicht-funktionaler Anforderungen eingesetzt werden kann. Die Internationale Organisation für Normung (ISO/IEC 25010) [ISO2011] z.B. definiert dabei folgende Qualitätsmerkmale:

Merkmal	Beschreibung des Systems
Funktionserfüllung	Das System soll Funktionen anbieten, um angegebene und implizierte Bedürfnisse zu erfüllen.
Effizienz	Funktionen des Systems sollen mit minimalem Verbrauch von Ressource ausgeführt werden.
Zuverlässigkeit	Funktionen sollen für einen bestimmten Zeitraum ausgeführt werden können.
Benutzbarkeit	Das System soll gut verstanden und gelernt werden können. Es soll auch attraktiv für den Benutzer sein.
Sicherheit	Informationen und Daten sollen geschützt werden, indem die Berechtigungsstufen für den Zugriff geregelt werden.

Merkmal	Beschreibung des Systems
Wartbarkeit	Änderungen an dem System sollen effektiv und effizient erfolgen.
Übertragbarkeit	Portierung von dem System oder dessen Komponenten soll effektiv und effizient erfolgen.
Kompatibilität	Das System soll Informationen mit anderen Systemen austauschen können.

Tabelle 1 Qualitätsmerkmale nach ISO/IEC 25010 [ISO2011]

Zu den nicht-funktionalen Anforderungen zählen außer den Qualitätsmerkmalen auch kulturelle, politische und rechtliche Anforderungen, welche je nach Projekt ebenfalls beachtet werden sollten.

Die nicht-funktionalen Anforderungen können oft auch antiproportional abhängig voneinander sein. So ist eine besonders sichere Anwendung von der Benutzbarkeit eher eingeschränkt, da man beispielsweise jedes Mal ein sehr langes Passwort mit einer Kombination aus verschiedenen Buchenstaben, Zahlen und Sonderzeichen eintippen muss, um sich einloggen zu können. Auch Effizienz und Wartbarkeit können in sich widersprüchlich sein, wenn z.B. Assembler Code in einer höheren Programmiersprache Umgebung eingebettet wird.

Neben der Quantifizierung spielt also auch die Priorisierung der nicht-funktionalen Anforderungen eine wichtige Rolle. Die Priorisierung sollte anhand konkreter Anwendungsfällen erfolgen. Bei der vertraglichen Gestaltung sollten die oben genannten Abhängigkeiten beachtet werden, damit die festgelegten Anforderungen auch vertragsgemäß realisiert werden können.

3 Rechtliche Einordnung

3.1 Werkvertrag und Abnahme

Bei vielen der IT-Projekte handelt es sich von der Herstellung einer individuellen Software oder Erweiterung eines bestehenden Systems. Als Vertragsform eignet sich demzufolge ein Werkvertrag, der in der Praxis fast immer eingesetzt wird. (vgl. BGH, 04.03.2010 - III ZR 79/09) [BGH2010]

Gem. § 631 BGB ist der Werkvertrag durch die Verpflichtung der Herstellung eines versprochenen Werkes vom Auftragnehmer und der Einrichtung einer vereinbarten Vergütung vom Arbeitgeber definiert. Der Auftragnehmer muss die zu erstellende Software gem. § 633 BGB frei von Sach- und Rechtsmängeln herstellen. Der Auftraggeber muss die vertragsmäßig hergestellte Werk Software gem. § 640 BGB abnehmen, dabei kann die Abnahme wegen unwesentlicher Mängel nicht verweigert werden.

3.2 Pflichtenheft und mittlerer Ausführungsstandard

Aufgrund der geltenden Gesetze ist es Interessen der beiden Vertragsparteien die Spezifikation möglichst genau zu beschreiben und diese in Vertragsgegenständen wie Pflichtenheften festzuhalten. Dennoch kommt es vor, dass das Pflichtenheft vergessen wird oder nur unvollständig oder nicht ausreichend detailliert vorliegt. In solchen Fällen sei: „*ein Ergebnis geschuldet, das unter Berücksichtigung des vertraglichen Zwecks des Programms dem Stand der Technik bei einem mittleren Ausführungsstandard entspricht*“. (vgl. BGH, 16.12.03 - *X ZR 129/01*) [BGH2003]

4 Methodik

4.1 Anforderungsanalyse

Die Anforderungsanalyse hilft Anforderungen des Auftraggebers systematisch zu sammeln und formal zu definieren. [Suz2012] Daher ist sie besonders wichtig, gerade wenn ein Urteil wie das im Kapitel 3.2 erwähnte Urteil vermieden werden sollte. Außerdem sind unvollständige Anforderungen nach der Chaos-Studie sehr häufig der Grund, woran Software Projekte scheitern. [Sta1994] Infolgedessen ist eine gute Anforderungsanalyse ein wichtiges Kriterium zum Projekterfolg. Damit werden auch Uneinigkeiten beim Projektergebnis vorgebeugt.

Ein wichtiger Schritt bei der Anforderungsanalyse ist die Anforderungsermittlung. Hierbei werden die Anforderungen vom Auftraggeber sowie weiteren Stakeholdern möglichst vollständig gesammelt. [Suz2012]

Bei der Ermittlung der Anforderungen sollte neben den Funktionalitäten der zu erstellenden Software auch die Qualität der Software spezifiziert werden. Viele dieser nicht-funktionalen Anforderungen werden nicht explizit erwähnt. Dennoch können einige Aussagen des Auftraggebers implizieren, welche Qualitätsmerkmale eine größere Rolle spielen.

Wenn der Auftraggeber beispielsweise aussagt, dass die Software von mehreren Tausenden Nutzern gleichzeitig genutzt wird, dann spielen Qualitätseigenschaften wie Skalierbarkeit eine nicht untergeordnete Rolle. Der Auftragnehmer sollte also bei unklaren Aussagen nachfragen, damit Uneinigkeiten bzw. Gerichtsurteile (vgl. Kapitel 3.2) später vermieden werden.

4.2 Maße für die Anforderungen

Die Nachprüfbarkeit der Erfüllung der Anforderungen an der zu erstellenden Software ist sehr wichtig für beide Vertragsparteien, da der Auftraggeber die hergestellte Software bei vertragsgemäßer Umsetzung abnehmen muss und der Auftragnehmer seine Vergütung dafür bekommt (vgl. Kapitel 3.1).

Dementsprechend muss die Qualität in messbaren Größen definiert werden, damit diese überprüft werden kann. Neben den Qualitätsanforderungen können auch weitere Anforderungen wie organisatorische oder technische Anforderungen eine Rolle spielen.

Folgende Beispiele veranschaulichen, wie bestimmte Anforderungen definiert und gemessen werden können:

- Technische Anforderungen (z.B. aus organisatorischen Gründen wie Lizenzen):
    - Das System muss mit der Programmiersprache Java 6 entwickelt werden und muss auf dem IBM WebSphere Application Server 7 laufen.
    - Die Daten des Systems müssen auf dem Datenbankmanagementsystem Oracle 11g gespeichert werden.

- Ergonomische Anforderungen (Benutzbarkeit):
    - Dialoge müssen gem. EN ISO 9241-110 gestaltet werden.

- Effizienz:
    - Das System muss jede Anfrage des Benutzers innerhalb von 5 Sekunden ausführen. Der Speicherbedarf darf 16 GB nicht übersteigen.

Weitere Anforderungen wie Erweiterbarkeit oder Wartbarkeit sind schwieriger zu messen, so dass man sie in der Regel nicht in einem kurzen Satz definieren kann. Hierbei existieren aber auch Maßnahmen, die diese Eigenschaften quantifizieren, welche im folgenden Kapitel erläutert werden.

4.3 Software Metriken

Eine Softwaremetrik bildet eine Software in einen Zahlenwert ab, dadurch kann man die Software formal vergleichen und bewerten.

Die IEEE Standard 1061 definiert Softwaremetrik wie folgt:

> *„Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen Zahlenwert abbildet, welcher als Erfüllungsgrad einer Qualitätseigenschaft der Software-Einheit interpretierbar ist. “*
>
> – *IEEE Standard 1061, 1998 [IEE1998]*

Die McCabe-Metrik, auch zyklomatische Komplexität genannt, z.B. messt die Komplexität eines Softwaremoduls. [Tho1983] Dabei wird unter anderem die Anzahl der IF Abfragen, also Kontrollfluss eines Moduls gemessen. Je höher diese Zahl ist, desto komplexer ist dieses Modul. Eine hohe Komplexität deutet auf schlechte Wartbarkeit hin, da eine erhöhte Anzahl an Tests geschrieben werden müsste, um alle Zweige der Logik im Modul zu überdecken.

Für die Erweiterbarkeit spielt unter anderem die Softwarearchitektur eine wichtige Rolle. Um die Architektur einer Software zu messen und zu bewerten, wird die Kopplung und Kohäsion der jeweiligen Komponente näher betrachtet. So hat eine Schichtenarchitektur bei einer sinnvollen Einteilung der Schichten eine starke Kohäsion und lose Kopplung. Will man in diesem Fall eine neue Komponente hinzufügen oder gewisse Komponenten austauschen, bedarf es bei einer guten Architektur nur wenig Aufwand.

Infolgedessen können auch Eigenschaften wie Wartbarkeit und Erweiterbarkeit quantifiziert werden, auch wenn es auf dem ersten Blick schwierig erscheint.

Der Betrieb und die Wartung haben bei vielen Softwaresystemen eine große Bedeutung. Deswegen sollten Wartbarkeit und Erweiterbarkeit ebenfalls in den Verträgen berücksichtigt und konkret festgehalten werden.

5 Fazit

Nicht vollständige Anforderungen ist der häufigste Grund, woran IT-Projekte scheitern. (vgl. Kapitel 4.1) Die Ursache der nicht vollständigen Anforderungen liegt vor allem an den Herausforderungen bei der Anforderungsanalyse.

Der Auftraggeber stammt meistens aus einem anderen Fachbereich und fokussiert bei den Anforderungen vor allem auf seinen gewünschten Funktionalitäten. Dabei werden die nicht-funktionalen Anforderungen an der zu erstellenden Software nicht bzw. nicht ausreichend explizit erwähnt. In solchen Fällen sollte der Auftragnehmer stets bei Unklarheiten nachfragen und auch die Anforderungen systematisch ermitteln, um diese quantifizieren zu können.

Nachdem die nicht-funktionalen Anforderungen vollständig gesammelt werden, ist es ebenfalls wichtig, diese messbar zu machen. Denn erst durch die Quantifizierung der Anforderungen können diese überprüft werden.

Viele dieser Anforderungen wie Effizienz oder technische Vorgaben können bereits mit einfachen Sätzen quantitativ definiert werden. Eigenschaften wie Wartbarkeit oder Erweiterbarkeit sind generell komplexer und können dann mit bestimmten Metriken gemessen und bewertet werden. (vgl. Kapitel 4.3)

Die quantifizierten nicht-funktionalen Anforderungen sollten wie die funktonalen Anforderungen auf alle Fälle auch in den Vertragsbestandteilen wie Pflichtenheft aufgenommen werden, damit Uneinigkeiten und Streitigkeiten im Nachhinein vermieden werden.

6 Quellenverzeichnis

[BGH2003] Bundesgerichtshof: *Urteil vom 16.12.2003 - X ZR 129/01.*, 2003.

[BGH2010] Bundesgerichtshof: *Urteil vom 04.03.2010 - III ZR 79/09.*, 2010.

[eWD2017] eWDG: Digitales Wörterbuch der deutschen Sprache. https://www.dwds.de/wb/Quantifizierung, zuletzt besucht am 28. Jan 2017.

[Fra2017] Sarre, Frank: Juristisches IT-Projektmanagement Skript: Allgemeines Vertragsrecht. https://www.sosy-lab.org/Teaching/2016-WS-JurPM/vorlesung-am-18.10.2016.pdf, zuletzt besucht am 28. Jan 2017.

[IEE1998] IEEE 1061: *Standard for a Software Quality Metrics Methodology.*, 1998.

[ISO2011] ISO/IEC 25010:2011: *Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models.*, 2011.

[Sta1994] Standish Group : *The CHAOS Report.*, 1994.

[Suz2012] Robertson, Suzanne; Robertson, James: *Mastering the Requirements Process: Getting Requirements Right.* 3rd. Auflage; Addison Wesley, 2012.

[Tho1983] MacCabe, Thomas J.: *Structured testing.* IEEE Computer Society Press, 1983.

[Wie2003] Wiegers, Karl E.: *Software Requirements.* 2nd. Auflage; Redmond: Microsoft Press., 2003.