Softwarequalität

Faktoren

Wir unterscheiden zwischen intern und extern Qualitätsfaktoren bei Software.

Die internen Qualitätsfaktoren können nur von Fachleuten bewertete werden. die externen Qualitätsfaktoren sind letztlich die wichtigsten Faktoren, da diese vom Benutzer während der Verwendung der Software wahrgenommen werden. Die externen Qualitätsfaktoren hängen jedoch fast immer von den internen Qualitätsfaktoren ab.

Faktoren die eine wichtige Rolle bei der Bewertung spielen, sind:

- Korrektheit
- Robustheit
- Erweiterbarkeit
- Wiederverwendbarkeit
- Kompatibilität
- Effizienz
- Portabilität
- Benutzerfreundlichkeit
- Funktionalität

Korrektheit: Erfüllt die Software ihre Aufgaben gemäß ihrer Spezifikation. Um Korrektheit zu erreichen, ist eine genaue Definition der Anforderungen erforderlich. Korrektheit kann in der Regel nur bedingt überprüft werden: „Wir garantieren die Korrektheit unseres Programms unter der Annahme, dass das „Fundament“ (Hardware wie auch Software) – auf denen unser Produkt basiert – korrekt sind“. Explizit, wenn der Prozessor falsche Ergebnisse berechnet, kann ihr Programm nicht korrekt sein. Siehe Wikipedia.

Robustheit: ist die Fähigkeit von Softwaresystemen, auf abnormale Bedingungen angemessen zu reagieren. Robustheit charakterisiert, was „außerhalb der Spezifikation“ geschieht und ergänzt die Korrektheit. Ist das „Fundament“ korrekt und Ihr Programm reagiert bei bestimmten Eingaben nicht, oder stürzt bei falschen Eingaben ab, ist die Robustheit nicht gegeben.

Die **Erweiterbarkeit** kennzeichnet die einfache Anpassung von Softwareprodukten an Änderungen der Spezifikation. Wichtige Prinzipien, um die Erweiterbarkeit zu erreichen: • Einfacher Entwurf, eine einfache Architektur lässt sich leichter anpassen. • Dezentralisierung, Autonome Module (Module mit minimaler Kopplung zu anderen Modulen lassen sich leichter ändern. In der Softwareentwicklung ist der Wandel allgegenwärtig.

Wiederverwendbarkeit ist die Fähigkeit von Software-Elemente öfter in vielen verschiedene Anwendungen wieder zu verwenden.

Kompatibilität ist die einfache Kombination von Softwareelementen mit anderer Software oder auch Hardware.

Portabilität kennzeichnet die einfache Übertragung von Softwareprodukten auf verschiedene Hardware- und Softwareumgebungen zum Beispiel die Portierung von Android auf iOS oder Portierung

von Windows auf Linux usw..

Effizienz ist die Fähigkeit eines Softwaresystems, so wenig Anforderungen wie möglich an Hardwareressourcen zu stellen, speziell in Bezug auf Rechenleistung, Festplattenspeicher, Arbeitsspeicher und die benötigte Bandbreite bei der Kommunikation. Versuchen Sie immer, „gute“ Algorithmen gegenüber „schlechten“ Algorithmen“ vorzuziehen.

Aber Geschwindigkeit ist nicht alles, es sei denn, wenn beide Algorithmen das gleiche, richtige Ergebnis liefern! Effizienz muss immer mit anderen Zielen in Einklang gebracht werden.

Funktionalität kennzeichnet den Umfang der Möglichkeiten, die ein System bietet. Vermeiden Sie Featurismus (einbauen von ungeforderten Features). Bleiben Sie mit den vorhandenen Funktionen konsistent, auch wenn Sie neue geforderte Features hinzufügen.

Benutzerfreundlichkeit ist der persönliche Aufwand, mit der Personen mit unterschiedlichen Hintergründen und Qualifikationen, bei der Benutzung Ihres Softwareproduktes empfinden.

Softwarefehler

Wie die Vergangenheit gezeigt hat, können Softwarefehler katastrophal sein. Beispiele hierfür sind:

> Therac-25, Menschen starben an einer Strahlungsüberdosis (1985). Ariane 5, das System aus Ariane 4 wurde wiederverwendet, aber die neuen Spezifikation wurde ignoriert (1996). Mars-Klima-Orbiter, Es gab keine Einigung über die verwendeten Einheiten, dadurch wurden metrische und zoll Angaben gemischt verwendet (1999).

Mangel an Softwarequalität ist aber auch, wenn ein System keine verständlichen Fehlermeldungen ausgibt, so dass der Benutzer weder weiss, ob er etwas falsch gemacht hat und was er falsch gemacht hat. Verständliche Fehlermeldungen erleichtern auch die Fehlersuche ungemein.

Beispielsweise auch:

> 2004 “Totalausfall” beim Lufthansa Buchungssystem, nach dem ein Softwareupdate aufgespielt wurde. Klar kann dies passieren und wenn der Support schnell handelt, kann der Schaden auch stark begrenzt werden. Doch peinlich wird es, wenn der selbe Fehler erneut ein Unternehmen in den Stillstand zwingt.
>
> Süddeutsche Zeitung 30.09.2009, 12:26:
>
> > “Totalausfall” bei Lufthansa Buchungssystem
> > Computerpanne bei Lufthansa: Mit Zettel und Stift musste die Lufthansa heute ihre Passagiere einchecken. Eine Computerpanne hatte den Check-In lahmgelegt. […]

Interne Qualitätsfaktoren

Wie schon C. Northcote Parkinson erklärte: […], wenn Sie etwas Komplexes bauen, werden nur wenige Leute mit Ihnen über Ihr Vorhaben streiten, da nur wenige Leute überhaupt verstehen, was Sie tun. Wenn Sie dagegen etwas bauen, das fast jeder bauen kann, dann hat auch jeder eine

Meinung […].

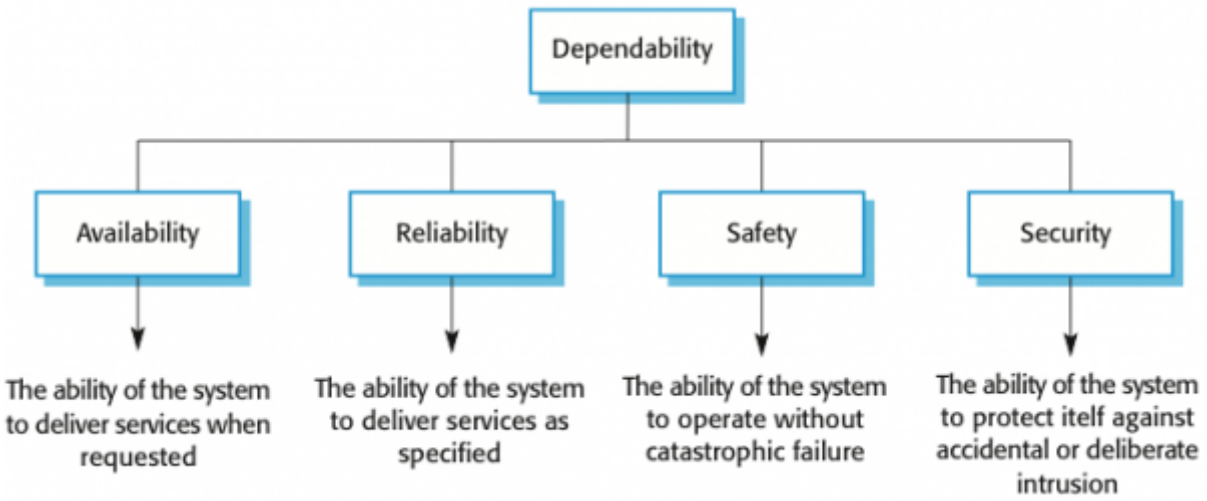
Oft steckt hinter schlecht lesbarem Code einfach nur Unwissenheit über die korrekte Umsetzung. Wenn man bei dem Grundsatz der Programierenung: „divide and conquer“ bleibt und große oder schwierige Probleme in kleine Teile bricht, wird dabei immer ein verständlicher und lesbarer Code entstehen.

Wenn Sie gute Programmierung erlernen möchten, gehen Sie auf Github und lesen Sie dort hochgeladen Code. Sie werden schnell sehen, was „guter“ und was „schlechter Code“ ist. Sie werden auch sehen, dass nicht nur der Code an sich zur Bewertung hinzugezogen wird, sondern auch die Dokumentation. Dort finden Sie auch Teile des Quellcodes für Comanche, Build 055. Dieser war Teil des benutzten Quellcodes in Apollo 11.

Quelle: educraftdiversions.org

Oft ist es auch nicht möglich, alle Eigenschaften der Softwarequalität zu verbessern, da diese in einem Widerspruch zu einander stehen.

- **Wartbarkeit**: Software sollte so geschrieben sein, dass sie sich an die sich ändernden Bedürfnisse der Kunden anpassen kann.
- **Effizienz**: Software sollte keine Systemressourcen verschwenden. Ausserdem zählt auch: Reaktionsfähigkeit, Bearbeitungszeit, Speicherauslastung usw. dazu
- **Benutzerfreundlichkeit**: Software muss für die vorgesehenen Benutzergruppe intuitiv verwendbar sein.
- **Zuverlässigkeit** oder **Verlässlichkeit**: Zuverlässige Software verursacht bei einem Systemausfall keinen physischen oder wirtschaftlichen Schaden.
- Weitere Eigenschaften wären dann projektspezifisch: **Reparaturfähigkeit**, **Langlebigkeit**, **Fehlertoleranz** …



Quelle: ayeshaasghar.uk Availability = Verfügbarkeit,
Reliability = Zuverlässigkeit,
Safety = Betriebssicherheit,
Security = Systemsicherheit

Statische Analyse-Tools

- FindBugs / SpotBugs, zur statische Analysen von Java-Bytecode.
- PMD, zur statische Analysen der AST mithilfe von Java-Visitors oder XPath basierten Regeln.
- CheckStyle, statische Analysen der AST mit Java-Visitors
- CheckerFramework, Statische Analysen mit pluggable Typen
- ConQAT, Code-Klonerkennung

Klassifizierung durch statische Analysewerkzeuge in richtig oder falsch Positiv. Dabei ist ein „true positiv“ Fehler die richtige Feststellung von etwas relevantem. Eine „false positiv“ Fehler ist eine Feststellung, die nur falsch ist. Einen ausführlichen Artikel finden Sie auf Wikipedia.

Zusammenfassung

Qualitätsfaktoren sind Projektspezifisch und die Qualität ist immer in Hinsicht auf das Projekt zu betrachten, da es unterschiedliche Prioritäten gibt. Dennoch ist meiner Meinung nach der wichtigste Aspekt: Verantwortung für seine Software zu übernehmen! Es gibt keine Ausreden. Wenn Sie ein System entwickeln, liegt es in Ihrer Verantwortung, es richtig zu machen. Übernehmen Sie diese Verantwortung. Mach es richtig oder mach es überhaupt nicht.

- Bei der Softwarequalität geht es nicht nur um die (interne) Qualität des Quellcodes.
- Softwarequalität bedeutet unterschiedliche Aspekte von verschiedenen Stakeholdern
- zur Erstellung von qualitativ hochwertiger Software ist die Sicht auf das gesamte Softwareprojekt erforderlich.