Universität Zürich[UZH]

Institut für Informatik

Martin Glinz

Software-Qualität – Ausgewählte Kapitel

Kapitel 9

Verlässlichkeit

Definition und Abgrenzung

Verlässlichkeit (Dependability) – Der Grad, in dem darauf vertraut werden kann, dass ein System zugesicherte Leistungen erbringt bzw. zugesicherte Eigenschaften hat.

Abzugrenzen von

- Zuverlässigkeit (reliability)
- Verfügbarkeit (availability)
- Informationssicherheit (security)
  - Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit, Verfügbarkeit
- Nutzungssicherheit (safety)
  - Sicherheitseigenschaften, Lebendigkeitseigenschaften

Bedrohungen

Verlust der Verlässlichkeit durch

- Fehler im System
  - Anforderungen korrekt interpretiert, aber fehlerhaft implementiert
  - Anforderungen falsch interpretiert
- Nicht erkannte, unerwünschte Systemeigenschaften
- Probleme in der Systemumgebung

Verlust geschieht

- Zufällig
- Fahrlässig
- In böser Absicht

Probleme in der Systemumgebung

- Fehler in der Systemumgebung
  - Fehler in Geräten oder Nachbarsystemen
  - Bedienfehler
  - Äußere Störereignisse

- Verletzung von Annahmen
  - Auftreten unerwarteter Fehler
  - Unerwartete Reaktion auf Ausgaben des Systems
  - Manipulationen durch Unberechtigte
  - Missbrauch durch Berechtigte

Maßnahmen

- Verhindern
- Erkennen und korrigieren
- Tolerieren
- Darlegen

- Kosten und Nutzen abwägen
- Verlässlichkeit ggf. nur für kritische Teile etablieren

Mittel

- Verlässlichkeit im Betrieb erreichen durch
  - Hinreichend häufigen Gebrauch
  - Selbstüberwachende Systeme
- Verlässlichkeit vor Inbetriebnahme erreichen durch
  - Analytisch, v.a. Test und Statische Analyse
  - Konstruktiv
    - Verifikation
    - Model Checking
    - Darlegung (Verlässlichkeitsfälle)
  - Rigorose Prozesse
- Vereinfachen durch Modularisierung

Test

- Systemtest: ungenügend zur Etablierung von Verlässlichkeit
- Mittel der Wahl:
  Zufallstest nach Benutzungsprofil
- Erlaubt statistische Prognosen
- Probleme: Ermittlung der Benutzerprofile
- Braucht sehr viele Testfälle
- Sicherstellen, dass die Systemumgebung mitgetestet wird (Ende-zu-Ende Tests)

Verifikation und Model Checking

- Verifikation
  - In der Regel nicht für ganze Systeme möglich
  - Erfasst nur das System, nicht seine Umgebung
  - Verifikationsfehler möglich

- Model Checking
  - Ganzer Zustandsraum in der Regel zu groß: keine Verifikation, sondern automatisierter Test
  - Erfasst nur das System, nicht seine Umgebung

Darlegung von Verlässlichkeit

- Festlegen der verlangten verlässlichen Eigenschaften
  - Je weniger, desto einfacher und kostengünstiger
- Bildung von Verlässlichkeitsfällen (dependability cases)
  - Konstruktion von Ende-zu-Ende-Argumenten für die verlangten Eigenschaften
  - unter Verwendung aller verfügbaren Techniken (einschl. Test, Verifikation, etc.)
  - Identifikation der erforderlichen Annahmen
  - Dokumentation der Annahmen (z.B. als Bestandteil einer Gebrauchsanweisung)
- Verlässlichkeitsfälle nicht nachträglich, sondern vorher erstellen
- Entwicklung an den Verlässlichkeitsfällen orientieren

Verlässliches Fundament notwendig

- Geeignete Programmiersprache
  - zum Beispiel mit starker Typprüfung
- Verlässliche Hardware
- Verlässliches Betriebssystem
- Verlässliche Kommunikationsinfrastruktur
- Aufbau auf verlässlichen Vorgängersystemen
  - aber: deren Verlässlichkeitsfälle revalidieren!

Aufwand für den Nachweis der Validität der Argumentationskette in den Verlässlichkeitsfällen steigt sonst ins Unermessliche

Literatur

D. Jackson (2009). A Direct Path to Dependable Software. *Communications of the ACM* **52**, 4. 78-88.

B. Nuseibeh, C. B. Haley, and C. Foster (2009). Securing the Skies: In Requirements We Trust. *IEEE Computer* **42**, 9 (September 2009). 64-72.

J. H. Saltzer, D. P. Reed, D. D. Clark (1984). End-to-End Arguments in System Design. *ACM Transactions on Computer Systems* **2**, 4 (November 1984). 277-288.