Douglas Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmidt (Hrsg.): INFORMATIK 2015
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2015

Grundlagen sicherheitsgerichteter Software-Entwicklung

Katja Stecklina[1] und Clemens Passeck[2]

Abstract: Um die Entwicklung eines sicheren Produktes erfolgreich abzuschließen sind neben technischen Aspekten und Anforderungen an die Software immer auch die Vorgehensweise und die Dokumentation der Entwicklungstätigkeiten für eine Zertifizierung bzw. Zulassung sehr wichtig. Dazu gehören die systematische Dokumentation von Anforderungen, eine systematische Implementierung und umfangreiche qualitätssichernde Aktivitäten wie das Testen und die Durchführung von Reviews. Neben den klassischen Software-Entwicklungsmethoden spielen seit längerer Zeit Vorgehensweisen wie Objektorientiertes Entwickeln, Agile Software-Entwicklung und Modellbasierte Entwicklung auch in der sicherheitsgerichteten Software-Entwicklung eine Rolle. Das Tutorial bietet einen branchenübergreifenden Einblick in die Anforderungen an die Entwicklung funktional sicherer Eingebetteter Software.

Keywords: Funktionale Sicherheit in der Software Entwicklung, Prozesse zur Entwicklung von sicherheitsgerichteter Eingebetteter Software

1 Einleitung

Das Thema Funktionale Sicherheit ist inzwischen in vielen Branchen auch für Softwareentwicklungsprojekte relevant geworden. Schon lange gibt es Anforderungen an Softwareentwicklungsprozesse aus Standards wie RTCA DO-178C, IEC 61508-1 und ISO 26262. Diese Standards werden regelmäßig weiterentwickelt und genügen immer höheren Ansprüchen an die Funktionale Sicherheit. Aktuell kann man beobachten, dass viele branchenspezifische Standards veröffentlicht werden, die sehr ähnliche Anforderungen beinhalten, beispielsweise ist seit einiger Zeit Modified Condition/ Decision Coverage als Abdeckungsmaß für geprüfte Softwareteile in RTCA DO-178C, DIN EN 61508-3 und auch in ISO 26262 gefordert. Aus einem Extrakt der wesentlichen Forderungen und Ziele dieser Standards erhält man einen Überblick über notwendige Arbeiten um für eine Software eine gegebenenfalls notwendige Zulassung zu erlangen. Im Folgenden wird auf diese wesentlichen Forderungen und Ziele eingegangen.

[1] Philotech Systementwicklung und Software GmbH, Kompetenzzentrum für Funktionale Sicherheit, Bahnhofstr. 30, 03046 Cottbus, Katja.Stecklina@philotech.de
[2] Philotech Systementwicklung und Software GmbH, Fachbereich Verifikation und Validierung, Bahnhofstr. 30, 03046 Cottbus, Clemens.Passeck@philotech.de

2 Ermitteln der zu erreichenden Sicherheitsstufe

Zu Beginn der Arbeiten an einer Software sollte bekannt sein, welche Regularien und Standards für die Entwicklung herangezogen werden müssen. Im Normalfall stehen die Regularien im Vorfeld fest.

Gemäß den jeweiligen Regularien wird die Stufe der Sicherheit für die Eingebettete Software basierend auf Analysen des Gesamtsystems festgelegt. Für die Funktionen, die das System erfüllt, werden dabei die Fehlfunktionen und deren Auswirkungen untersucht. Je nach Schwere der Folgen, teilweise mit Berücksichtigung weiterer Faktoren wie Kontrollierbarkeit und Eintrittswahrscheinlichkeit von bestimmten Situationen, wird für die Funktion eine Stufe der Sicherheit bestimmt. In DIN EN 61508-1 beispielsweise werden diese „Safety Integrity Level“ genannt.

Die jeweilige Stufe der Sicherheit wird dann gemäß bestimmter Standard-abhängiger Regeln der Software und deren Funktionen zugeordnet. Auf Basis dieser Stufen werden die Maßnahmen bestimmt, die während der Software-Entwicklung ergriffen werden müssen um den Standard zu erfüllen. Man geht dann davon aus, dass eine Software, die die Vorgaben erfüllt, hinreichend sicher ist. Dieses Konzept wird im Luftfahrtbereich „Development Assurance“ genannt [RTa].

3 Der Lebenszyklus der Software-Entwicklung

Ein wesentlicher Aspekt der Development Assurance ist das Festlegen eines für das jeweilige Softwareentwicklungsprojekt passenden Lebenszyklus. Normalerweise besteht ein solcher Lebenszyklus aus mehreren Phasen, beispielsweise Anforderungsdefinition, Design, Implementierung, statische Analyse und dynamisches Testen. Für jede dieser Phasen werden Ein- und Ausgangsinformationen festgelegt, welche Aktivitäten innerhalb der Phasen stattfinden und welche Übergangskriterien zwischen den Phasen gelten müssen.

Bestimmte Anforderungen müssen in den zu definierenden Phasen des Lebenszyklus abgebildet werden. Diese Anforderungen werden in der Regel spezifisch für die Stufe der Sicherheit festgelegt und betreffen häufig die überprüfenden Aktivitäten. Beispielsweise müssen gemäß RTCA DO-178C für jede Software-Anforderung im Pflichtenheft eine adäquate Menge von Testfällen existieren, die auf der Zielhardware unter möglichst realistischen Einsatzbedingungen durchgeführt werden müssen. Während der Definition des Lebenszyklus müssen diese Anforderungen berücksichtigt werden.

Die Phasen selbst müssen mitnichten strikt in Reihenfolge oder einmalig durchgeführt werden. Iterative inkrementelle Ansätze sind gängig. In vielen Projekten werden diese

Phasen mehrfach oder parallel durchgeführt. Wichtig ist, dass die Übergangskriterien erfüllt sind bevor Aktivitäten begonnen werden und dass die Anforderungen an das Konfigurationsmanagement jederzeit erfüllt sind. Hierzu gehört zum Beispiel, dass nach der Änderung einer Anforderung je nach Fortschritt die Folgen der Änderung analysiert und dokumentiert werden müssen. Eine Anforderungsänderung kann zum Beispiel eine Teständerung oder Implementierungsänderung nach sich ziehen, oder eine Überprüfung anderer Anforderungen auf Konsistenz kann erforderlich werden.

Es ist sogar möglich, innerhalb des Projektes mehrere unterschiedliche Lebenszyklen für verschiedene Teile der Software zu definieren. Der Zukauf eines Teils der Software erfordert einen ganz anderen Lebenszyklus als eine Neuimplementierung oder die Wiederverwendung von Modulen aus anderen Projekten.

Die Definition eines oder mehrerer Lebenszyklen für ein Projekt ist nicht immer einfach. Neben den Regularien müssen auch erwartete Aufwände, Werkzeuge, Kompetenzen der Entwickler, Zeitpläne und andere Faktoren berücksichtigt werden.

4 Einbindung von speziellen Techniken und Sonderfällen

In der Praxis stellt sich heraus, dass ein Softwareentwicklungsprozess mit der gängigen Abfolge von Anforderungsdefinition, Design, Implementierung in Verbindung mit Reviews und Testen sehr gut beherrschbar ist. Nicht selten jedoch ist diese gängige Abfolge für das Projekt bzw. Teilprojekt nicht zutreffend. Oft kommt es beispielsweise vor, dass Teile der Software zugekauft oder extern entwickelt werden, Bibliotheken oder alte Softwareteile integriert werden müssen oder Werkzeuge benutzt werden sollen, die bestimmte Prozessschritte automatisieren. Oft treten in solchen Fällen Probleme auf, zum Beispiel dass Anforderungen zu wiederverwendeten Softwareteilen fehlen oder dass das Vertrauen in die automatisierenden Werkzeuge nicht vorhanden ist.

Die Behörden und Regularien bieten für solche Ausnahmen entweder direkt Lösungsansätze an, z.B. geht RTCA DO-331 auf Anforderungen an Modellbasiertes Entwickeln ein. Ansonsten ist es in der Regel ratsam, mit der zulassenden Behörde oder Institution frühzeitig das Gespräch zu suchen, wie mit den entsprechenden Forderungen in diesen Sonderfällen umzugehen ist.

5 Zusammenfassung

Der Ansatz „Development Assurance“ oder vergleichbaren Konzepten für die Produktentwicklung soll sicherstellen, dass das Verhalten der Produktsoftware mit hinreichender Wahrscheinlichkeit keine Gefahr für Menschen darstellt. Innerhalb eines geeigneten zu definierenden Lebenszyklus werden alle Anforderung an den Prozess abgebildet, die je nach Sicherheitsstufe erforderlich sind.

Literaturverzeichnis

[DIa] DIN EN 61508-1: Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/programmierbarer elektronischer Systeme – Teil 1: Allgemeine Anforderungen (IEC 61508-1:2010)

[DIb] DIN EN 61508-3: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme – Teil 3: Anforderungen an Software (IEC 61508-3:2010)

[RTa] Radio Technical Commission for Aeronautics: RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification. Washington: RTCA, 2011 (RTCA/DO-178C)

[RTb] Radio Technical Commission for Aeronautics: RTCA DO-331: Model-Based Development and Verification Supplement to DO-178C and DO-278A. Washington: RTCA, 2011 (RTCA/DO-331)

[ISO] International Organization for Standardization: ISO 26262:2011: Road vehicles – Functional safety. Geneva, 2011