ACX GmbH

DO-178C

Eine Einführung

Inhalt

Hintergrund 2
Software Level 2
Entwicklungsprozesse 3
Weitere Techniken 5
Partioning 6
Multiple-Version Dissimilar Software 6
Safety Monitoring 6

Die DO-178C (Software Considerations in Airborne Systems and Equipment Certification) ist ein Standard zur Softwareentwicklung. Der von der RTCA entwickelte und von der EUROCAE unter dem Namen ED-12B übernommene Standard soll die Sicherheit und Zuverlässigkeit von, in der Luftfahrt verwendeter sicherheitskritischer Software, sicherstellen. Sowohl die europäische EASA als auch die US-amerikanische FAA fordern für den Softwareentwicklungsprozess die Einhaltung des Standards.

Die folgenden Erläuterungen zeigen die Kernpunkte dieses Standards und verdeutlichen ihre Maßnahmen, Techniken und Prozesse für die Entwicklung von Software für den Einsatz in der Luftfahrt.

Nicht jede Software ist gleichermaßen kritisch einzuordnen. Der Ausfall eines Autopiloten kann größere Auswirkungen haben als der Ausfall der Kabinenbeleuchtung. Hierzu werden die Wahrscheinlichkeit des Ausfalls und dessen Auswirkungen einem von 5 Sicherheitsleveln zugeordnet. Je größer die Auswirkungen des Ausfalls eines Systems sind, desto auswendiger ist der in der DO-178C geforderte Entwicklungsprozess. Dieser definiert insbesondere Prozesse und Artefakte, um einen Nachweis der Entwicklungen belastbar einzufordern. Ein besonderer Schwerpunkt liegt auf der Einführung von Teststrategien. Außerdem werden verschiedene Techniken diskutiert, welche die Zuverlässigkeit eines Avioniksystems erhöhen.

Hintergrund

Seinen Ursprung fand die Urversion DO-178 durch die FAA im Jahre 1980 in den USA. Triebfeder für die Entwicklung dieses Standards war die drastische Zunahme von Softwarefunktionalitäten in Flugzeugen und den damit entstehenden Risikofaktoren für die Sicherheit.

Um die erforderliche Zertifizierung zu erhalten, müssen die Hersteller belegen, dass der implementierte Softwarecode alle entsprechenden Anforderungen erfüllt, dass der Code keine Passagen enthält, die unbestimmte Einflüsse auf die Software haben können und dass der Code komplett und umfassend gemäß den Bestimmungen getestet wurde.

Software Level

Je nach möglichen Auswirkungen eines unerkannten Fehlers wird die Software in fünf unterschiedliche Software Levels unterteilt:

Level A	Verhindert den sicheren Weiterflug oder die Landung.
Level B	Führt zu einer hohen Belastung für die Besatzung das Flugzeug zu handhaben und möglicherweise zu schweren oder tödlichen Verletzungen der Insassen.
Level C	Führt zu zunehmender Arbeitsbelastung für die Crew und verursacht möglicherweise Unbehagen und leichte bis mittelschwere Verletzungen der Insassen.
Level D	Führt zu geringer Ausfallgefahr. Von der Besatzung ist eine erhöhte Aufmerksamkeit gefordert.
Level E	Hat keine nachteiligen Auswirkungen auf die Sicherheit des Fluges oder die Flugzeuginsassen.

Die Levels beziehen sich auf die Bedeutung des Systems für das Flugzeug. Flugsteuerung, Navigation, und alle Fly-by-Wire-Systeme sind flugkritisch und erfordern DO-178C Level A Zertifizierung. Entertainment-Systeme fallen dagegen am unteren Ende des Spektrums nur unter Level E.

Der entsprechende Software Level wird durch das System Safety Assessment bestimmt. Der Level eines Softwaresystems muss früh in der Entwicklung festgelegt werden, da er vor allem erhebliche Auswirkungen auf die Architektur der Software hat. Folgende Entscheidungen können davon betroffen sein:

- Die Programmiersprache
- Coding Styles und zulässige Paradigmen
- Die Möglichkeit der Abbildung auf Anforderungen (Requirements Traceability)
- Testbarkeit einzelner Units
- Verwendung von Threads und Prozessen

Entwicklungsprozesse

Die DO-178C beschreibt drei Hauptprozesse für den Software-Entwicklungsprozess:

- Planung
- Entwicklung
- Test

Für jeden dieser Prozesse müssen Planungsdokumente entsprechend der Sicherheitseinstufung des Projektes erarbeitet werden. Die folgende Liste zeigt das Grundgerüst der benötigten Dokumente:

- Plan for Software Aspects of Certification (PSAC)
- Software Quality Assurance Plan
- Software Configuration Management Plan
- Configuration Control Procedures
- Software Code Standard
- Software Design Standard
- Software Requirements Standard
- Software Development Plan
- Software Verification Plan
- Source, Executable Object Code, SCI and SECI
- Software Design Document
- Software Requirements Document
- Traceability
- Test Cases and Procedures
- Verification Results
- Quality Assurance Records
- Configuration Management Records
- Problem Reports
- Software Accomplishments Summary

Der Planungsprozess definiert das Vorgehen während des gesamten Entwicklungsprojekts bis zum Abschluss der Zertifizierung. Das Hauptdokument, welches in der Planungsphase

entsteht, ist der Plan for Software Aspects of Certification (PSAC). Er referenziert i.d.R. alle anderen Planungsdokumente und soll den gesamten Entwicklungsprozess in Kurzform wiedergeben. Wichtig sind dabei alle sicherheitskritischen Elemente, sowie der Ablauf des Entwicklungsprozesses insgesamt.

Der PSAC dient während der ganzen Projektlaufzeit als Leitfaden für die Entwickler des Systems und hilft bei der Abstimmung und Kommunikation mit der Zulassungsbehörde. Außerdem werden mit der Behörde Abnahmekriterien und Termine vereinbart.

Die DO-178C schreibt kein Vorgehensmodell zur Softwareentwicklung vor, macht aber strenge Qualitätsvorgaben. Während der Planung muss also ein Softwareentwicklungsprozess definiert werden, der es ermöglicht die erstellte Software auf höchstem technischem Niveau zu testen bzw. durch angewendete Verfahren und Richtlinien Fehler auszuschließen bzw. aufzudecken.

Grundsätzlich fordert die DO-178C eine anforderungsbasierte Entwicklung (Requirements Engineering). Diese werden auf drei Ebenen abverlangt:

System Level Requirements	beschreiben Funktionen und Eigenschaften der zu entwickelnden Software nach außen, also zum Anwender hin.
High Level Requirements	beschreiben die grundsätzlichen Anforderungen und Designentscheidungen an das System um das System Level Requirements zu erfüllen, welchem sie zugeordnet worden. (Was soll getan werden?)
Low Level Requirements	beschreiben wie genau die beschriebene Funktionalität des High Level Requirement, welchem sie zugeordnet sind, implementiert werden soll. (Wie soll etwas getan werden?)

Der Software Verifikations-Prozess (Reviews, Analyse und Test) stellt einen zentralen Aspekt innerhalb der DO-178C dar. Sie definiert ihn als Kombination von Bewertungen (Reviews), Analysen und Tests. Die erforderlichen Tests, Prüfungen und Nachweise hängen vom definierten Software Level ab. Aufgabe ist es Fehler in der Software zu erkennen und zu entfernen. Ein Fehler ist ein Verhalten der Software, welches von den Anforderungen abweicht bzw. nicht beschrieben ist. Erkannte Fehler werden entfernt und betroffene Entwicklungsschritte wie Design, Programmierung, sowie der Test müssen wiederholt werden.

Software Reviews und Analysen sind ein Mittel zum Nachweis der korrekten Umsetzung von während der Planung festgelegten Richtlinien für zum Beispiel Anforderungen, Tests, Gestaltung des Quellcodes usw., von funktionalen Anforderungen die nicht über einen automatischen Test geprüft werden können, aber auch von nichtfunktionalen Anforderungen. Reviews und Analysen definieren meist über Checklisten wie genau ein Aspekt der entwickelten Software bzw. des Software Moduls im Detail zu untersuchen.

Die Anforderungsebenen High-Level-Requirements und Low-Level-Requirements werden untersucht, um festzustellen, ob Fehler oder Unstimmigkeiten in den Anforderungskatalog eingeflossen sind. Jedes Requirement wird überprüft, um sicherzustellen, dass es klar und eindeutig die Anforderungen an das Gesamtsystem (System Level Requirements) erfüllt und nicht im Widerspruch zu anderen Anforderungen steht. Es wird auch geprüft, ob

Anforderungen fehlen um das zu entwickelnde System vollständig zu beschreiben. Um dies sicherzustellen muss jedes Requirement eindeutig auf Anforderungen des Gesamtsystems zurückzuverfolgen sein (Traceability). Ein funktionales Requirement muss außerdem prüfbar, ein nichtfunktionales Requirement muss bewertbar sein und die Darstellung muss einheitlichen, frei definierbaren Standards genügen.

Die verwendeten Algorithmen werden auf Genauigkeit und deterministisches Verhalten überprüft.
Die Software-Architektur wird überprüft und analysiert, um Fehler und Widersprüche aufzudecken. Alle Architektur-Komponenten müssen auf Konsistenz und Kompatibilität mit der Embedded Hardware überprüft werden.

Der Source Code wird überprüft und analysiert um Fehler aufzudecken und die Konformität zu den Software Code-Standards zu gewährleisten. Je nach festgelegter Coding Guideline wird der Sprachumfang der verwendeten Programmiersprache beschränkt um Laufzeitfehler von vornherein auszuschließen. Der Code muss mit der Embedded Hardware kompatibel sein und alle Low-Level-Requirements abdecken. Datenfluss und die Ablaufsteuerung soll der Software-Architektur entsprechen. Der Code soll leicht verständlich sein und darf eine bestimmte Komplexität nicht übersteigen.

Die Software wird getestet, um sicherzustellen, dass sie den Anforderungen entspricht und um alle möglichen Ursachen, die zu einem nicht definierten Fehlerzustand oder Absturz führen können zu eliminieren.

Testfälle im Normalbereich (Normal Range Test Cases) sollten zu jedem Low-Level-Requirement entwickelt werden um gültige Eingabesequenzen und Grenzwerte zu testen. Die Testfälle sollen dabei möglichst alle Zustandsübergänge während des normalen Betriebs abdecken. Darüber hinaus sieht die DO-178C Testfälle für ungültige Eingabesequenzen vor (Robustness Test Cases). Wichtig in diesem Zusammenhang ist die Reaktion von arithmetischen Funktionen für ungültige Eingabewerte und die Überprüfung auf Überlaufbedingungen.

Die Test-Coverage-Analyse soll durchgeführt werden, um sicherzustellen, dass es mindestens einen Testfall für jede Anforderung an die Software gibt und dass sowohl im Normalbereich als auch außerhalb des normalen Eingabebereichs Testfälle definiert wurden.

Die Structural-Coverage-Analyse stellt sicher, dass jede Code-Sequenz mindestens einmal bei der Durchführung der Tests durchlaufen wurde. Gelingt dies nicht, so müssen die verbliebenen Code-Sequenzen analysiert und ggf. entfernt werden. Es besteht auch die Möglichkeit, nicht durchlaufene Code-Sequenzen gewollt zu deaktivieren. Geschieht dies nicht, so spricht man von sogenanntem „Dead Code“. Dieser wird im normalen Zustand des Systems zwar nicht ausgeführt, in einem ungewollten Fehlerzustand befürchtet man hier jedoch ein ungewolltes und unvorhersehbares Verhalten des Systems.

Weitere Techniken

Die DO-178C gibt zusätzliche Empfehlungen für die Architektur von sicherheitskritischen Systemen, um die Zuverlässigkeit des Systems zu erhöhen. Drei von ihnen werden im Folgenden kurz beschrieben. Im jeweiligen Projekt muss bewertet werden, ob und welche Architekturen dazu geeignet sind die Sicherheit und Zuverlässigkeit zu erhöhen.

Partioning

Partitioning trennt funktional unabhängige Komponenten des Systems, so dass bei einem Ausfall einer dieser Komponenten die anderen Komponenten unabhängig weiterhin funktionieren können. Eine geringe Kopplung zwischen den Komponenten und auch den Daten von den verwendeten Komponenten ist der Schlüssel für ein erfolgreiches Partitioning.

Partitioning kann eine reine Softwarelösung, eine Hardwarelösung oder eine Kombination von Hardware und Software Partitioning sein. Wenn zusätzliche Software erforderlich ist, um Partitioning zu ermöglichen, muss sie entsprechend dem höchsten Software Level der zu partitionierenden Komponenten dokumentiert und getestet werden.

Multiple-Version Dissimilar Software

Die Technik des Multiple-Version Dissimilar Software, häufig als N-Version Programmierung bezeichnet, beschreibt die Bereitstellung redundanter Komponenten, die die gleiche Funktionalität bieten, aber in verschiedener Art und Weise umgesetzt wurden.

Diese Strategie ermöglicht redundante Funktionalität indem die Implementierung der einen Komponente keinen Einfluss auf die Implementierung auf ihre Schwesterkomponente hat. Sind jedoch die Eingabedaten beschädigt, dann kann auch die N-Version Programmierung das System nicht schützen.

Ein weiteres Problem besteht in der Lösung von Konflikten, wenn eine Schwesterkomponente ausfällt oder andere Werte zurückliefert. Das System stellt im ersten Schritt nur fest, dass es einen Fehler gibt. Es ist jedoch nicht klar, welche Komponente fehlerfrei gearbeitet hat. Aus diesem Grund wurde das Safety Monitoring als typische Technologie in Systemen der Luftfahrt eingeführt.

Safety Monitoring

Safety Monitoring bedeutet, dass das Verhalten einer Komponente permanent von einer zweiten Software Komponente überwacht wird. Im Problemfall soll das Versagen der Komponente frühzeitig erkannt und Schutzmaßnahmen eingeleitet werden.

Hierzu wird das Verhalten der überwachten Komponente beobachtet und dessen Schlüssigkeit bewertet. Diese Funktionalität kann durch den Einsatz von Software, Hardware oder einer Kombination aus Beiden implementiert werden (Watchdog, Built-In-Test, etc.).