

DO-178B

Software considerations in airborne systems and equipment certification

Dennis Walter

Deutsches Forschungszentrum für Künstliche Intelligenz, Bremen
Sichere Kognitive Systeme

Vortragrunde, 2008-12-11

*Some **plans** were made and rice was thrown
A house was built, a baby born
How time can move both fast and slow
amazes me*

(C. Oberst, "I Believe In Symmetry")

Agenda

- 1 Übersicht
- 2 Systembezogene Aspekte
- 3 Software-Lebenszyklus
- 4 Planungsprozess
- 5 Entwicklungsprozess
- 6 Verifikationsprozess
- 7 Konfigurationsmanagementprozess
- 8 Produkte des Software-Lebenszyklus

Was ist die DO-178B?

- DO-178B besteht aus einer Menge von *Richtlinien* zur Sicherstellung, dass Software mit Luftfahrtanforderungen (*airworthiness req.*) in Einklang ist.
- Diese Richtlinien kommen in drei Formen:
 - ▶ Ziele (*“objectives”*) für SW-Lebenszyklusprozesse
 - ▶ Aktivitäten und Designüberlegungen, um diese Ziele zu erreichen
 - ▶ Beschreibung von Nachweisen, die Erfüllung der Ziele bestätigen
- DO-178B bezieht sich lediglich auf den SW-Lebenszyklus; der Systemlebenszyklus wird nur bezüglich seiner Schnittstelle zum SW-Lebenszyklus dargestellt. Organisatorische Aspekte werden nicht behandelt.
- Angegebene Richtlinien sind per se nicht gesetzlich bindend

Einordnung

Die DO-178B ist ein international anerkannter und der meistverwendete Standard zur Entwicklung von sicherheitsbezogener Software in der Luftfahrt.

- Veröffentlicht: 1992 (davor: DO-178A in 1985, DO-178 Ende 1970er)
- Entworfen von RTCA, Inc. (gemeinnützige Einrichtung, USA, mit internat. Konsortialmitgliedern)
- Referenziert z.B. in FAA Anforderungsdokumenten zur Zertifizierung von SW zum Einsatz in der Luftfahrt (FAA Order 8110.49 "*Software Approval Guidelines*")
- DO-248B als "FAQ" zur DO-178B
- Findet auch Anwendung in militärischen Projekten
- DO-178C geplant für 2009! Größte Neuerung: Berücksichtigung von modellbasierter SW-Entwicklung und neuen Verifikationsmethoden

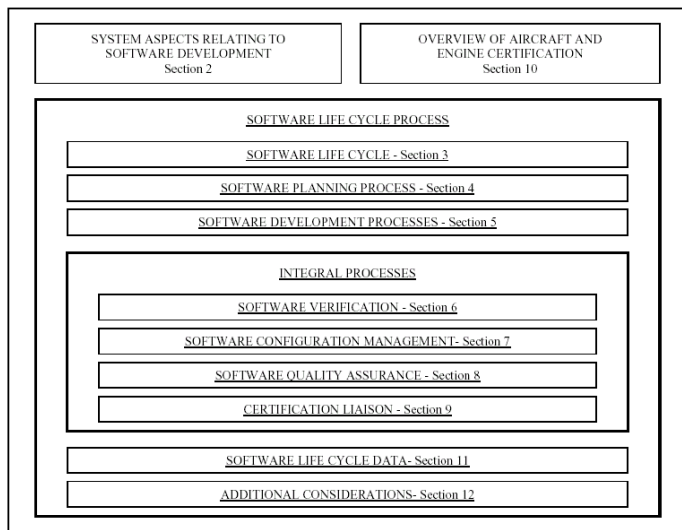


FIGURE 1-1
DOCUMENT OVERVIEW

6 Prozesse

Prozesse in der DO-178B

- Planung
 - Entwicklung
 - Verifikation
 - Konfigurationsmanagement
 - Qualitätssicherung
 - Zusammenarbeit ("*liaison*") mit der Zertifizierungsbehörde
-
- Prozesse sind weiter gefasst als V-Modell-artiger Prozess aus 61508
 - Entwicklungsprozess noch weiter untergliedert
 - Ziele und Produkte der Prozesse sind tabellarisch in Annex A aufgelistet

Softwareanforderungen leiten sich von Systemanforderungen ab. Daher beinhaltet das Dokument Angaben zu den Schnittstellen zwischen Systemaspekten und der Softwareentwicklung.

- Festlegung der Ausfallkategorien und SW-Level (nächste Folie)
- Systemarchitektur, die Entwicklung konformer SW ermöglicht:
 - ▶ Partitionierung
 - ▶ Mehrkanalige, nichtidentische SW
 - ▶ Überwachung: Safety Monitoring
- Weitere Aspekte (User-modifiable & field-loadable SW, ...)

Failurekategorien und SW-Level

Failure Condition	Software Level
Catastrophic: Sicherer Flug und Landung nicht mehr möglich	A
Hazardous/Severe-Major: Beeinträchtigung von Crew und Maschine, so dass (1) Große Reduktion von Sicherheitsmargins/Funktionalität, (2) Crew kann Aufgaben nicht mehr zuverlässig vollständig ausführen, (3) Schwere/tödliche Verletzungen von Besatzung/Passagieren	B
Major: Einschränkung von Funktionalität, Crewbelastung, mögl. Verletzungen	C
Minor: Leichte Einschränkungen, z.B. Flugplanänderungen, Unbequemlichkeiten für Passagiere	D
No Effect (i.F. nicht berücksichtigt)	E

Software-Lebenszyklus

Unter einem Software-Lebenszyklus wird (hier) die Menge von Prozessen sowie deren Abfolge und gegenseitige Abhängigkeiten verstanden, die zur Erstellung eines SW-Produkts ausgeführt werden.

Wichtigster Punkt (des 2 Seiten langen Abschnitts):

Transitionskriterien

- Entscheiden, ob ein Prozess begonnen/wiederaufgenommen werden darf
- Feedback von einem Prozess zum anderen

Offene Formulierungen dienen der Flexibilisierung: verschiedene Prozessmodelle können zur Anwendung kommen, sofern hier genannte Kriterien erfüllt sind

Ziele

- SW-Entwicklungs- und Integritätsprozesse sind definiert
- Transitionskriterien, Beziehungen und Abfolge von Prozessen sind definiert
- Umgebung des Software-Lebenszyklus (Methoden, Werkzeuge) ist definiert
- Software-Entwicklungsstandards sind definiert
- Softwarepläne genügen Anforderungen dieses Dokuments
- Softwarepläne sind aufeinander abgestimmt

Planungsprozess

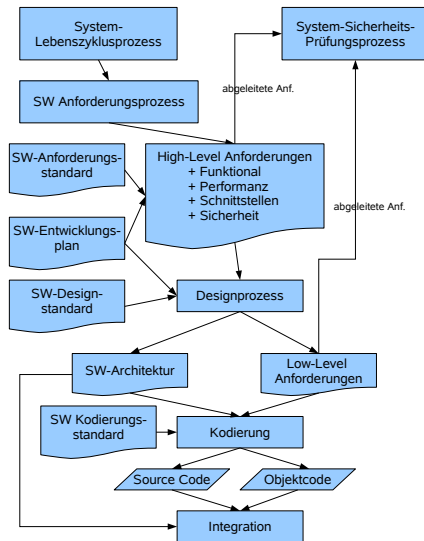
Pläne

- Plan for Software Aspects of Certification
- Software Development Plan
- Software Verification Plan
- Software Configuration Management Plan
- Software Quality Assurance Plan

Weitere Aspekte

- Compileroptimierungen: Strukturelle Abdeckung des Objektcodes
- Planung ist essentiell für erfolgreiche Zertifizierung

Übersicht Softwareentwicklungsprozess



Entwicklungsprozess

- Es wird kein Entwicklungsprozess (z.B. V-Modell) vorgeschrieben, sondern lediglich Ziele für Teilprozesse vorgegeben \rightsquigarrow paralleler Ablauf von Prozessen möglich
- Möglichkeit der Revision vorangegangener Entwicklungsschritte: Feedback bei Fehlern/Inkonsistenzen
- Berücksichtigung der Möglichkeit, direkt aus HL-Anforderungen Code abzuleiten
- Rückverfolgbarkeit: Alle Verfeinerungsschritte müssen auf Anforderungen der höher gelegenen Ebenen basieren

Verifikation \neq Testen

Verification is not simply testing. Testing, in general, cannot show the absence of errors. As a result, the following subsections use the term “verify” instead of “test” when the software verification process objectives being discussed are typically a combination of reviews, analyses and test.

- Verifikation ist eine technische Beurteilung und Prüfung der Ergebnisse sowohl des SW-Entwicklungsprozesses als auch des Verifikationsprozesses selbst
- Definiert durch: Software Planungsprozess & Software Verification Plan

Ziele der Verifikation

Zweck des Verifikationprozesses ist die Aufdeckung und der Report von Fehlern. Behebung dieser Fehler erfolgt innerhalb des Softwareentwicklungsprozesses.

Ziele

- SW-Systemanforderungen wurden in SW-High-Level Anforderungen umgesetzt, die diese erfüllen
- High-Level Anforderungen wurden in Architektur und Low-Level Anforderungen umgesetzt, die diese erfüllen
- Architektur und Low-Level Anforderungen wurden in Source Code umgesetzt, der diese erfüllt
- Der ausführbare Objektcode erfüllt *die* SW-Anforderungen
- Die verwendeten Mittel, um diese Ziele zu erreichen sind technisch korrekt und vollständig für den SW-Level

Reviews und Analysen

Definition (Review)

Ein Review liefert eine qualitative Beurteilung der Korrektheit. Es kann z.B. aus einer Inspektion eines Produkts anhand einer Checkliste bestehen.

Definition (Analyse)

Eine Analyse liefert einen wiederholbaren Nachweis der Korrektheit. Bspw. kann eine Analyse aus einer detaillierten Untersuchung der Auswirkungen der Funktionalität, Performanz, Rückverfolgbarkeit oder Sicherheit einer Softwarekomponente bestehen, sowie ihrer Beziehung zu anderen Komponenten.

Arten von Reviews und Analysen

- Kompatibilität mit Anforderungen (System, High-/Low-Level)
- Konsistenz
- Kompatibilität mit Zielarchitektur/-rechner
- Verifizierbarkeit
- Standardkonformität
- Integrität von Partitionen (Architektur)
- Rückverfolgbarkeit
- Algorithmische Aspekte: Genauigkeit und Verhalten von A., insb. im Bereich von Unstetigkeiten
- Fehlerfreiheit von Source Code: Stack, Auflösung und Overflow von Fixpunktarithmetik, Ressourcenverwendung, Ausführungszeitabschätzungen, Ausnahmebehandlung, uninitialisierte Variablen, Interrupts

Software Testprozess

Drei Testarten:

- Hardware/Software Integrationstests
- Software Integrationstests
- Low-Level Tests

Ziele

- Tests sollten auf SW-Anforderungen basieren
- Testfälle sollten korrekte Funktionalität verifizieren und Bedingungen möglicher Fehler festlegen
- Anforderungsabdeckungsanalyse
- Strukturelle Abdeckungsanalyse

Software Testprozess (Abb. 6.1)

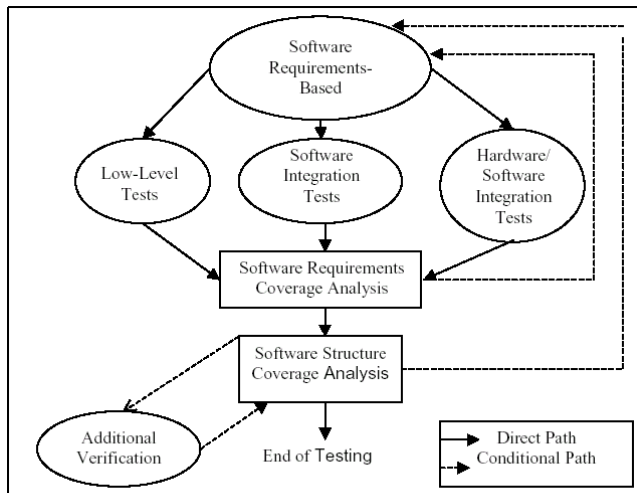


FIGURE 6-1
SOFTWARE TESTING PROCESS

Auswahl von Testfällen

Normaler Wertebereich

- Äquivalenzklassen und Randfälle für reelle/ganzzahlige Variablen
- Mehrfache Iterationen für zeitabhängige Funktionen
- Alle möglichen Zustandsübergänge in normalem Operationsmodus
- Analyse/Review von logischen Ausdrücken & strukturelle Abdeckung

Robustheit

- Ungültige Eingabebereiche
- Systeminitialisierung unter abnormen Bedingungen
- Mögliche Fehlerarten von Eingabeströmen
- Out-of-range Werte für berechnete Schleifenzähler
- Behandlung von Zeitüberschreitungen
- Arithmetischer Überlauf bei zeitabhängigen Funktionen
- Unerlaubte Zustandsübergänge

Modified Condition/Decision Coverage

Definition (MC/DC)

Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by varying just that condition while holding fixed all other possible conditions.

- Stärkstes in der Praxis angewandtes Coverage-Kriterium
- Gefordert für Software Level A (B erfordert Decision Coverage)

Modified Condition/Decision Coverage

Beispiel

A	B	C	$A \wedge (\neg B \vee C)$
0	0	0	0
1	0	0	1
1	1	0	0
1	1	1	1

- Erfordert *im Idealfall* $N + 1$ Testfälle für eine N -stellige Boole'sche Funktion!
- Kompromiss zwischen Test aller möglichen Kombinationen und hoher Aufdeckungswirkung

Fehleraufdeckung durch anforderungsbas. Testen (Bsp.)

Hardware/Software Integration

- Interruptbehandlung
- Probleme mit Bus und anderen Ressourcen
- Probleme mit Hardware-Speicherverwaltung

Software Integration

- Variableninitialisierung
- Parameterübergabe
- Numerische Auflösung

Low-Level Tests

- Schleifenoperationen
- Logische Inkonsistenzen
- Behandlung falscher Eingabewerte

Konfigurationsmanagementprozess

- Mehr als nur “Wir benutzen svn”: Betonung liegt auf organisatorischer, nicht auf technischer Lösung
- Verantwortlichkeiten, Nachvollziehbarkeit, Auswirkungskontrolle
- Ziele des Konfigurationsmanagementprozesses:

Identifikation der K.-Einheiten	Produkt-Baselines
Rückverfolgbarkeit	Problemlberichte
Änderungskontrolle: Integrität	Änderungskontrolle: Verfolgung
Änderungsrevision	Sicherung gg. unbefugte Änderung
Datensicherung	Verfügbarkeit

Produkte des Software-Lebenszyklus

- Anforderungen an Eindeutigkeit, Vollständigkeit, Konsistenz, Klarheit, Verifizierbarkeit, Rückverfolgbarkeit, Änderbarkeit
- Zusammengesetzt aus *Plänen, Standards, Anforderungen, Design, Code, Verifikationsergebnissen, Berichten, Qualitätssicherungsdokumenten*
- Zentrales Dokument: *Plan for Software Aspects of Certification*
- Siehe Abschnitt 11 des Dokuments (*"Software Life Cycle Data"*)

Vergleich zur IEC 61508

- IEC 61508 hat den Status einer Europäischen Norm; DO-178B hat bezüglich internationaler Zulassungsrichtlinien eine Art Referenzstatus
- DO-178B ist konkreter bezüglich zu erstellender Plandokumente
- IEC 61508 unterscheidet Funktionale Anforderungen und Integritätsanforderungen; DO-178B kennt nur High-/Low-Level Anforderungen
- IEC 61508 nennt Sätze von anwendbaren Methoden und formuliert allgemeine Anforderungen; DO-178B ist wesentlich konkreter bezüglich technischer Anforderungen, sieht weniger Methoden vor
- DO-178B stellt Formale Methoden nicht auf gleiche Stufe wie Testen (*"12.3: Alternative Methods"*); IEC 61508 setzt Testschwerpunkt, erwähnt andere Methoden als teilweise gleichwertig

Zusammenfassung

- DO-178B ist *der* Standard zur Entwicklung von Software in der Luftfahrt
- Im Gegensatz zur IEC 61508 ist er sehr “griffig” und gut lesbar, weil konkret
- Planungsorientierung: Nach Fertigstellung der Pläne sind zu verwendende Tools, Methoden und Zeitplan festgelegt
- Testorientierung: Keine/kaum formale Methoden, und wenn: eher modellbasierte Ansätze, statische Analysen
- Anforderungen an Software Level A extrem hoch: Unabhängigkeit der Verifikation, MC/DC, Rückverfolgbarkeit

Ende

