

Verlässliche Echtzeitsysteme

Fallstudien

Peter Ulbrich

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
www4.informatik.uni-erlangen.de

07. Juli 2014



- Wie sind **kommerzielle verlässliche Systeme** aufgebaut?
 - Welche **Fehler** gilt es zur Laufzeit zu tolerieren?
 - Welche **Mechanismen** werden für die Fehlertoleranz eingesetzt?
 - Welche Maßnahmen stellen die **Korrektheit der Implementierung** sicher?

- Betrachtung zweier **Fallstudien**:
 - Sizewell B: **primäres Reaktorschutzsystem**
 - Airbus A320/A330/A340: **digitale „Fly-by-Wire“-Systeme**

 - **Schwerpunkt** der Fallstudien:
 - Grundverständnis der Funktion dieser Systeme
 - struktureller Aufbau der Systeme hinsichtlich Fehlertoleranz
 - Verifikation der eingesetzten Software



- 1 Überblick
- 2 Sizewell B
 - Überblick
 - Reaktorschutzsystem
 - Softwareverifikation
- 3 Airbus: Fly-by-Wire Flight Controls
 - Flugsteuerung
 - Flugsteuerung A320/A330/A340
 - Architekturoptimierung
 - Softwareverifikation@AIRBUS
- 4 Zusammenfassung





Standort Suffolk, UK

Betreiber EDF Energy

Erbauer u. a.

- Westinghouse
- Framatome (Areva)
- Babcock Eneergy
- GEC-Alsthom

Entwurf: 1980-82

Bau: 1988-95

Laufzeit 2035

Leistungsdaten

- Elektrisch: 1195 MW
- Thermisch: 3479 MW

(Quelle: John Brodrick)



- **Zweck:** Durchführung einer **Reaktorschnellabschaltung (RESA)**
 - (engl. *reactor emergency shutdown, reactor trip, SCRAM*)
 - SCRAM = „Safety Cut Rope Axe Man“
 - manuelle Notabschaltung des ersten Kernreaktors 1942
 - falls ein **unsicherer Reaktorzustand** festgestellt wird
- **Funktionsweise** der Schnellabschaltung
 - Einschließen der **Steuerstäbe** (engl. *control rod*) in den Reaktorkern
 - in Druckwasserreaktoren werden diese von oben eingeschossen
 - Normalbetrieb: Magnete/Motoren pressen sie gegen vorgespannt Federn
 - zusätzlich: Einleiten von **Neutronengiften**, z. B. Borsäure

↪ Einfangen freier Neutronen, **Stoppen der Kettenreaktion**

 - Reaktorleistung reduziert sich auf die **Nachzerfallswärme** (engl. *decay heat*)
 - diese beträgt ca. 5% der thermischen Leistung ↪ ca. 174 MW (Sizewell B)
- **Sicherheitsanforderung: fail-operational**

↪ den **sicheren Zustand** (engl. *fail-safe*) nimmt der Reaktor ein



- Ausschluss von **Gleichtaktfehlern**

↪ verursacht durch Fehler im Entwurf oder der Implementierung

- ☞ **Diversitärer Aufbau** des Schutzsystems

- **primäres Schutzsystem** (engl. *primary protection sys., PPS*)

- basierend auf digitaler Sicherheitsleittechnik

- Überwachung von **Reaktorparametern**

- Neutronenfluss im Reaktordruckbehälter
- ^{16}N -Gehalt im Primärkühlkreislauf

- Überwachung der **Steuerstäbe**

- **Reaktorinstrumentierung** (engl. *reactor instrumentation*)

- **Stromkreisunterbrecher** (engl. *circuit breakers*) ↪ SCRAM

- **sekundäres Schutzsystem** (engl. *secondary protection sys., SPS*)

- basierend auf diskret aufgebauten, analogen Schaltungen



- **Zuverlässigkeitsanforderung:** Toleranz eines ausgefallenen Kanals
 - auch wenn ein Kanal aktuell gewartet wird
 - Wartungen/Tests während des Betriebs sind unumgänglich
 - der Reaktor wird nur zur Revision und zur Wiederbefüllung heruntergefahren
 - ↪ diese Revisionsintervalle betragen typischerweise 18 Monate
 - ↪ vierkanaliger, redundanter Aufbau des PPS
 - außerdem wird sichergestellt, dass maximal ein Kanal gewartet wird
- **zulässige Ausfallwahrscheinlichkeiten** des PPS
 - „failure upon demand“ $\leadsto f/d$
 - Ausfall eines einzelnen Kanals: $10^{-3}f/d$
 - insgesamt (das redundante System aus vier Kanälen): $10^{-4}f/d$
 - Ausfallwahrscheinlichkeit: $10^{-5}f/year$
- ☞ **jeder unsichere Zustand** führt zur RESA
 - auch wenn das PPS **nicht mehr aktiv in der Lage ist**, dafür zu sorgen
 - z. B. weil der Strom für die Schutzsysteme komplett ausfällt
 - ↪ **Passivität der Systeme** hat Auslösung der Sicherheitsfunktionen zur Folge



■ 4-fach redundante Sicherheitsleittechnik

- Redundanz umfasst jeweils Sensorik, Berechnung und Aktuatoren
- ↪ die Replikation umfasst also den **kompletten Kontrollpfad**
 - im Umfeld von Sizewell B sprach man „Guardlines“
- einzelne Redundanzen sind **räumlich separiert**
 - Aufstellorte der Kontrollrechner, Kabelkanäle, Stromversorgung, ...
- ↪ Vermeidung von **Gleichtaktfehlern durch Umwelteinflüsse**

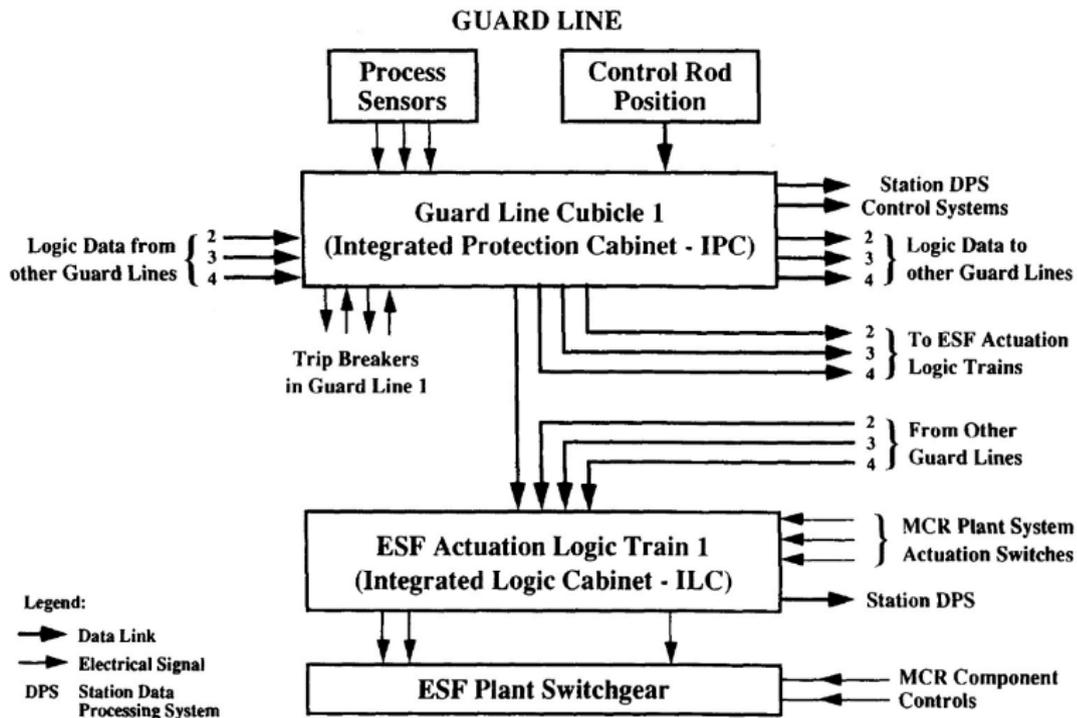
■ **unabhängige Arbeitsweise** der einzelne Replikate

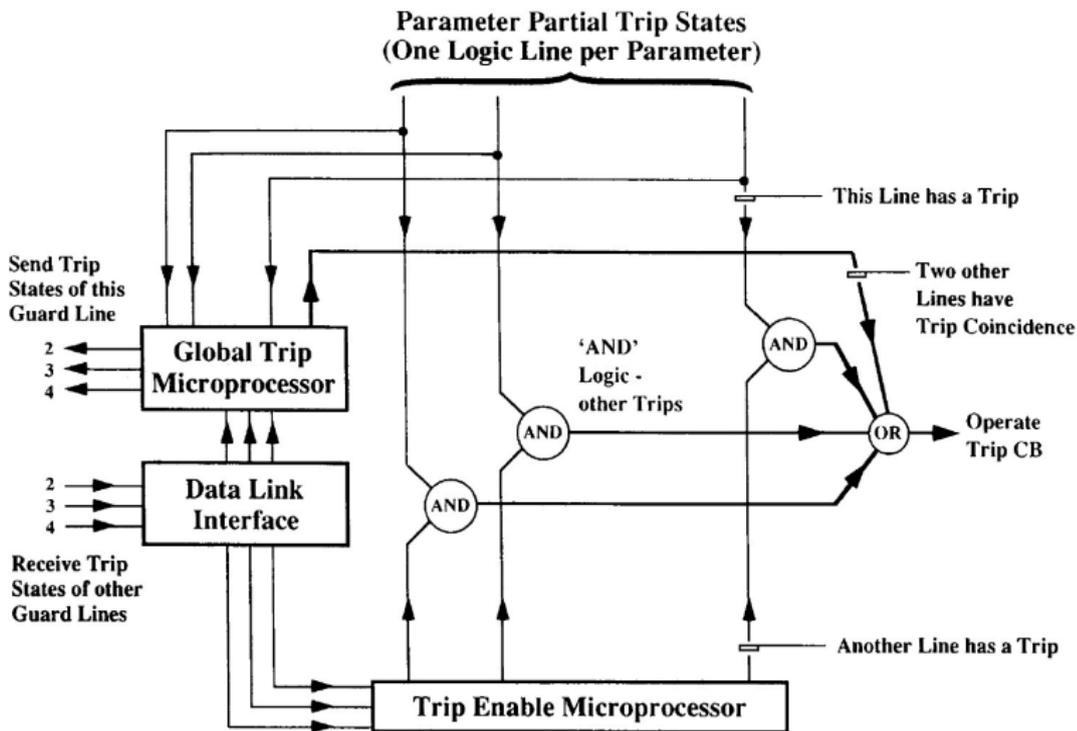
- sie bestimmen eigenständig ob eine RESA vonnöten ist
- ↪ Durchführung der RESA wird durch **Mehrheitsentscheid** ermittelt
 - jedes Replikat führt den Mehrheitsentscheid selbst durch
 - die Logik des Mehrheitsentscheids bezieht sich auf einen Wahrheitswert
- ↪ Implementierung durch einen dedizierten Schaltkreis
- ↪ hierfür notwendige Kommunikation erfolgt über **optische Medien**
 - keine gegenseitige **elektrische Beeinflussung**
 - keine Störungen durch **elektromagnetische Interferenz**



Eine „Guardline“ des primären Schutzsystems

Quelle Grafik: [4]





- Verifikation und Validierung bestand aus verschiedenen Aktivitäten:
 - Engineering Confirmatory Analysis NNC Ltd.
 - Begutachtung (engl. *review*) relevanter Entwicklungsdokumente
 - Anforderungen/Spezifikationen für System/Code, Quellcode und -daten
 - Independent Design Assesment Nuclear Electric
 - Überprüfung der Systemanforderungen in Systementwurf/-spezifikation
 - Einbeziehung von Software-Entwurf und -Spezifikation
 - MALPAS Analysis TA Consultancy Services Ltd.
 - formale Verifikation der Softwareimplementierung mit MALPAS
 - Object/Source Code Comparison Nuclear Electric
 - Nachweis der Äquivalenz zwischen Binär- und Quellcode mit MALPAS
 - Dynamic Testing Rolls Royce and Associates Ltd.
 - Durchführung von ca. 55000 zufällig erzeugten Testfällen
- gesamter geschätzter Aufwand: 250 Mannjahre
 - in etwa derselbe Aufwand wurde bereits von Westinghouse investiert



- Entwicklung durch „Royal Signals and Radar Establishment“
 - Forschungseinheit des britischen Verteidigungsministeriums
 - Stationierung in Malvern (Worcestershire) \leadsto Namensgebung
- besteht aus folgenden Analysewerkzeugen

Kontrollflussanalyse \mapsto Kontrollflussgraph ...

- Schleifen, Ein-/Ausstiegspunkte, Reduzierbarkeit, ...

Datenflussanalyse \mapsto erreichende Definitionen, ...

- Verwendung nicht initialisierter Daten, nie geschriebene Ausgaben

Informationsflussanalyse (engl. *program dependency graph*)

- Daten- und Kontrollflussabhängigkeiten von Ausgabevariablen

semantische Analyse \mapsto symbolische Ausführung

- funktionale Zusammenhänge zwischen Ein- und Ausgaben

Einhaltung von Vor- und Nachbedingungen

- (engl. *compliance analysis*)



- **zu prüfen:** Softwareimplementierung des PPS
 - Implementierung in PL/M-86 und ASM86 bzw. PL/M-51 und ASM51
 - umfasst insgesamt ca. 100.000 *Lines of Code*
 - ca. 40.000 Zeilen für einen Hauptprozessor, ca. 10.000 bei Hilfsprozessoren
 - Anwendung, Betriebssystem, Kommunikation, Selbsttest, ...
- **Referenz:** Anforderungs- und Entwurfsdokumente
 - „Software Design Requirements“ (SDR)
 - abstrakte Beschreibung der von der Software zu erbringenden Funktionalität
 - „Software Design Specification“ (SDS)
 - architekturelle Umsetzung der funktionalen Anforderungen
 - enthält detaillierte Information zur Funktion einzelne Softwarekomponenten
 - beschreibt bereits alle Programmvariablen, sowie Ein- und Ausgaben
- **Ablauf:** Verifikation erfolgt Prozedur für Prozedur (engl. *unit proof*)
 - aufgerufene Prozeduren werden durch geeignete Platzhalter ersetzt
 - beginnend bei Blattprozeduren



- MALPAS verwendet eine **eigene Zwischensprache: MALPAS IL**
 - für den PL/M-86-Code wurde ein eigener Übersetzer entwickelt
 - ↪ **Problem:** MALPAS IL unterstützt anders als PL/M-86 **keine Zeiger**
 - Implementierung von z. B. OUT- und INOUT-Parametern durch Zeiger
 - ↪ **Lösung: Dereferenzierung** per Zeiger angesprochener Objekte
 - **Kodierrichtlinien** ↪ eingeschränkte Verwendung von Zeigern
 - ↪ Dereferenzierung erfolgt **größtenteils automatisiert**, **teilweise manuell**
- **semantische Analyse** ↪ Extraktion funktionaler Zusammenhänge
 - Ergebnis ist der mathematische Zusammenhang: Eingabe \mapsto Ausgabe
 - ↪ manueller Abgleich mit den Anforderungen/der Spezifikation
- Formulierung von **Vor- und Nachbedingungen** in MALPAS IL
 - Ansatz: primäre Quelle SDR, Verfeinerung mithilfe von SDS
 - ↪ schwierig wegen unterschiedlich detaillierter SDR/SDS
 - Analyse war **sehr mühsam** ↪ alternative Formulierungen waren oft nötig
 - ungünstiger, schwer zu vereinfachender Ausdruck ließ Analyse scheitern
 - ↪ Neuformulierung „wies der algebraischen Vereinfachung den Weg“



- **Problem:** korrekte Formulierung von Vor-/Nachbedingungen
 - 1 standardisierter Analyseprozess (ISO 9001)
 - 2 detaillierte Vorgehensbeschreibung für die Durchführung (ca. 200 Seiten)
 - 3 detaillierte Protokollierung der Analyse
 - Eingabe für die MALPAS-Analyse und ihre Ergebnisse
 - für jede Analyse wurden vorgefertigte Formulare ausgefüllt
 - Ableitung der math. Spezifikation, Interpretation der Ergebnisse, ...
 - 4 umfangreiche gegenseitige Begutachtung (engl. *peer-review*)
 - Einhaltung des Prozesses, Verständnis des PPS erweitern
 - Überprüfung von Terminierungsbeweisen, Termersetzungsregeln, ...
- **Ergebnisse:** Abweichungen von der Spezifikation
 - diese wurden kommentiert und kategorisiert
 - ↪ Lieferung von insgesamt ca. 2000 Kommentaren an Nuclear Electric
 - Kategorie 1 mögliche Fehlfunktion im PPS ↪ keine
 - Kategorie 2 Änderungen in Anforderungen/Spezifikation ↪ ca. 40%
 - Kategorie 3 nicht-kritische Änderungen am Quelltext ↪ ca. 8%
 - Kategorie 4 keinerlei Änderung erforderlich ↪ ca. 52%



Äquivalenz von Quell- und Binärcode [5]

Traue Nichts und Niemandem, ... auch nicht dem Übersetzer!

- **Problem:** Passt der Binärcode auch zum Quellcode?
 - Was hilft der korrekteste Quellcode, wenn der Übersetzer fehlerhaft ist?
 - bewiesenermaßen korrekte Übersetzer existierten damals nicht
 - nimmt man Assemblierer und Binder dazu, ist das auch heute noch so
 - Rekonstruktion des Quellcodes aus dem Binärcode ist nicht möglich
 - ~> kein Vergleich „originärer vs. rekonstruierter Quellcodes“



Idee: man trifft sich in der Mitte ~> MALPAS IL

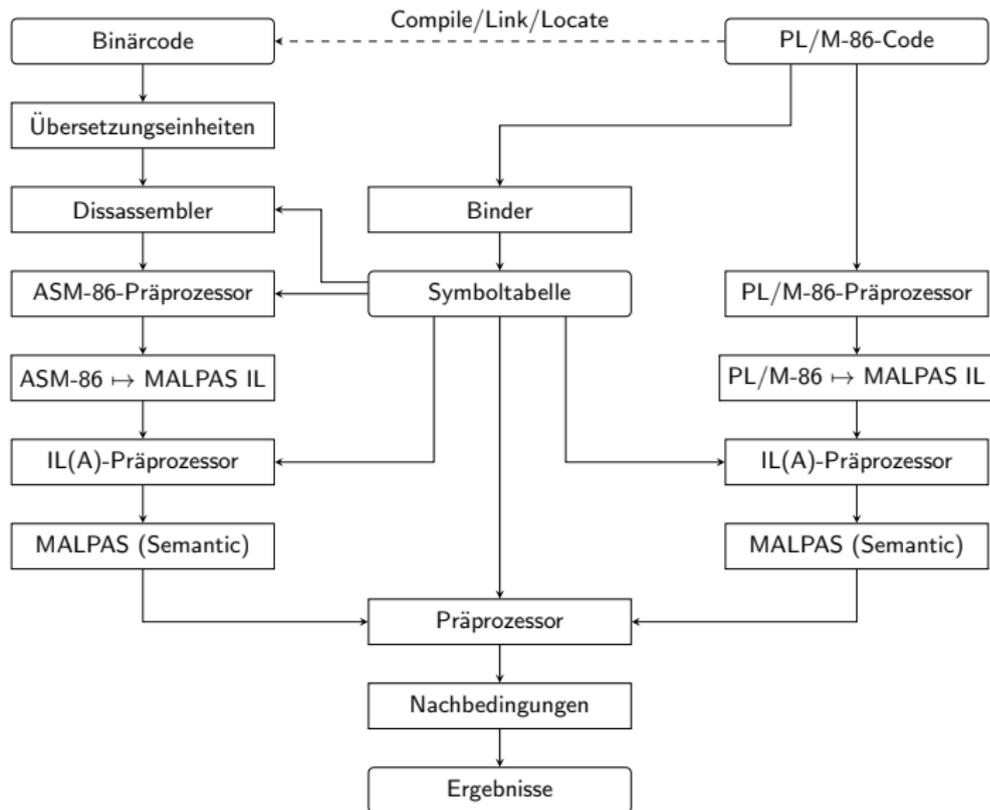
- Übersetzer „PL/M-86 ~> MALPAS IL“ existiert bereits
- Übersetzer „Binärcode ~> MALPAS IL“ entwickelt man noch
 - Rekonstruktion der Übersetzungseinheiten, Disassemblierung, ...
- Vergleich \mapsto Verifikation der Nachbedingungen mit MALPAS
 - Quellcode ~> Extraktion von Nachbedingungen
 - Binärcode ~> Extraktion der Implementierung

~> **zu zeigen:** die Implementierung erfüllt die Nachbedingung

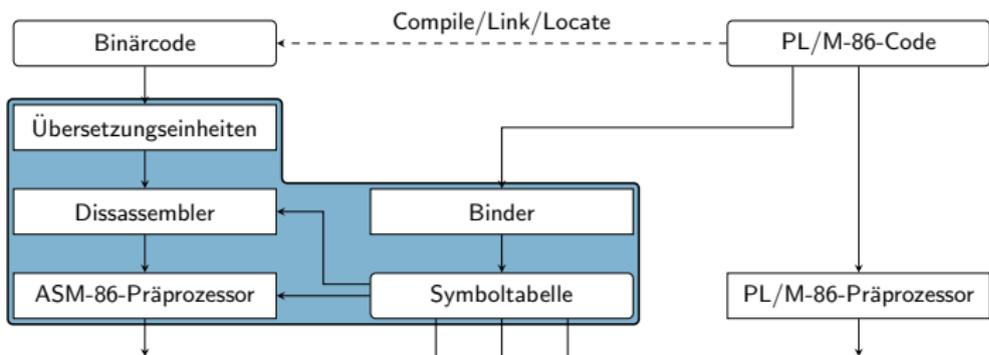
~> Quell- und Binärcode sind identisch



Ablauf des Vergleichs: Quell- vs. Binärcode



Ablauf des Vergleichs: Quell- vs. Binärcode



■ Rekonstruktion von Text- und Datensektionen

↪ ursprüngliche Übersetzungseinheiten

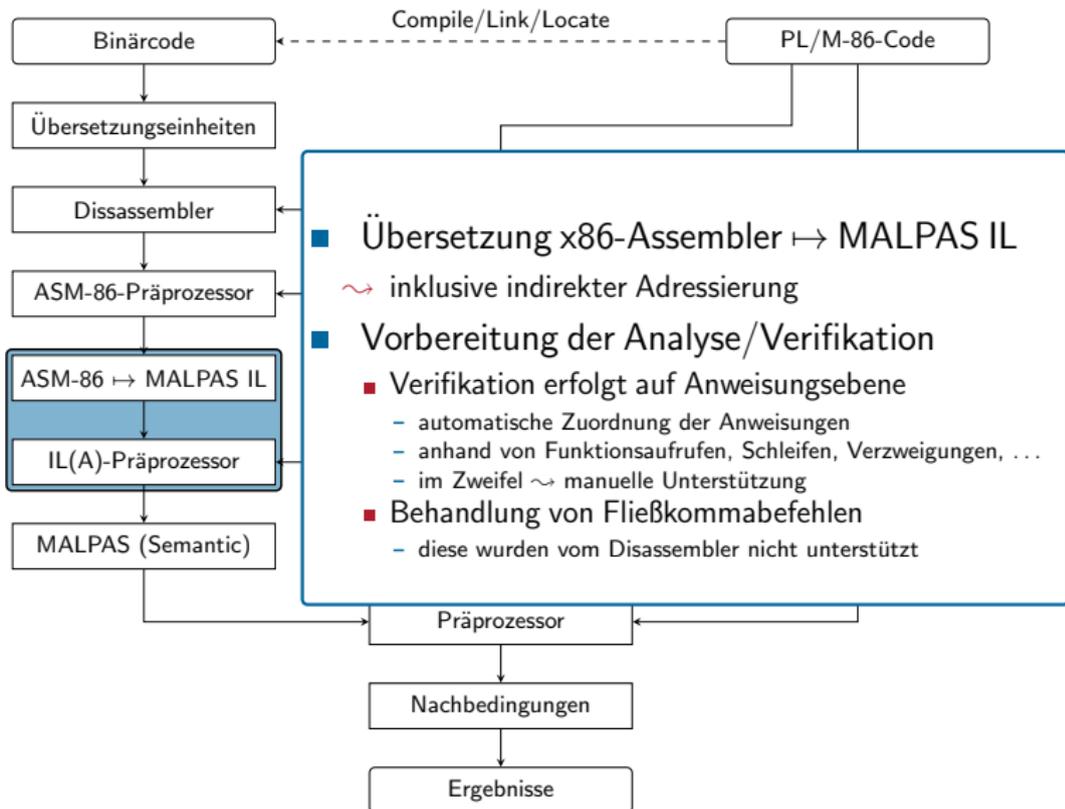
■ Dissassemblierung des Binärcodes

■ Aufbereitung des Assemblercodes

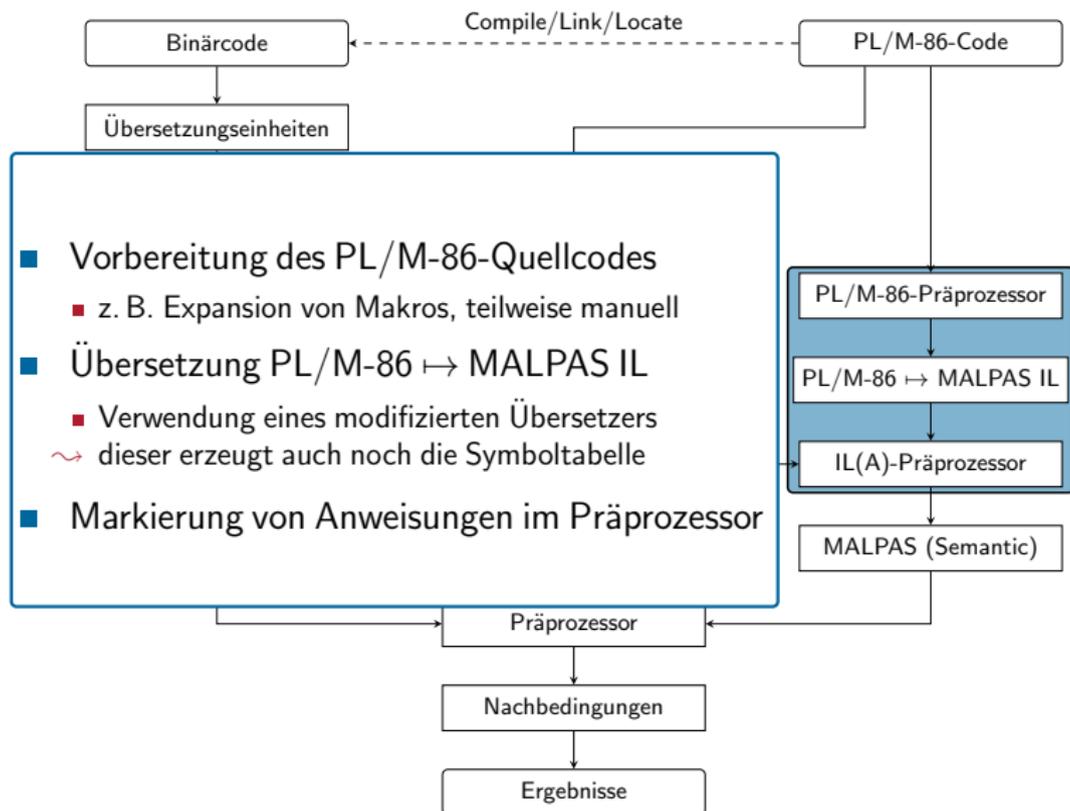
- unterstützt durch eine aus dem Quelltext bestimmte Symboltabelle
- Bestimmung von globalen Variablen aus Adressen
- Bestimmung von Funktionsparametern
- Einfügen von Variablendeklarationen



Ablauf des Vergleichs: Quell- vs. Binärcode

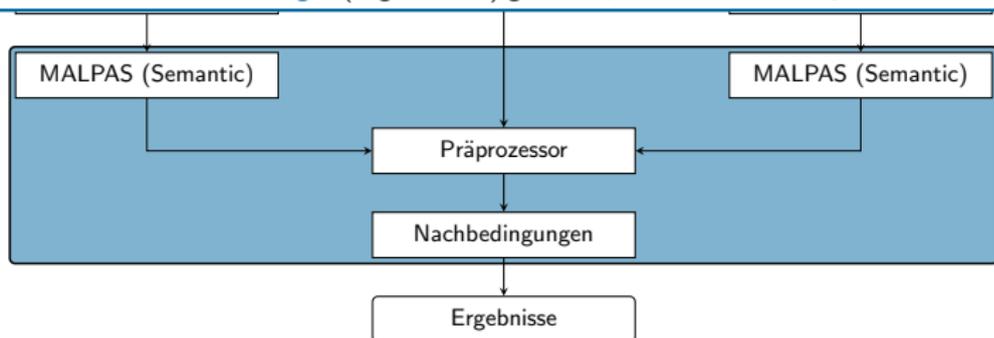


Ablauf des Vergleichs: Quell- vs. Binärcode



Ablauf des Vergleichs: Quell- vs. Binärcode

- funktionalen Zusammenhänge zwischen Ein- und Ausgabe
 - ~ Eingabe für die Prüfung der Nachbedingungen
 - MALPAS vergleicht nicht direkt den erzeugten MALPAS IL-Code
 - ~ es stellt die extrahierten math. Zusammenhänge gegenüber
- Formulierung des Verifikationsproblems in MALPAS IL
 - Eliminierung verbliebener, problematischer Konstrukte
 - Speicherreferenzen durch indirekte Adressierung, Registerzuweisungen, temporäre Variablen
 - Zuordnung der Anweisungen durchführen: ASM-86 ↔ PL/M-86
 - ASM-86-Anweisungen werden zu Prozedurimplementierungen in MALPAS IL
 - PL/M-86-Anweisungen werden zu Nachbedingungen in MALPAS IL
- Überprüfung der Nachbedingungen durch MALPAS
 - wurden keine **Bedrohungen** (engl. *threats*) gefunden, waren Binär- und Quellcode identisch



- **insgesamt: 11 Abweichungen** zwischen Binär- und Quellcode [3]
 - eine davon stellte sich als ernsthafter Defekt des Übersetzers heraus
 - Ergebnisse wurde nicht offiziell veröffentlicht, „sickerten aber durch“

■ Bewertung des Ansatzes

Generalisierbarkeit \leadsto Portierung für andere Programmiersprachen

- Ansatz \leadsto allgemein gehalten, Implementierung \leadsto sprachabhängig
- PL/M ist eine sehr einfache Sprache und erleichtert die Verifikation
 - komplexere Sprachen könnten dieses Vorhaben erschweren
 - Optimierungen wie das Ausrollen von Schleifen etc. gar unmöglich machen

Automatisierbarkeit war in weiten Teilen gegeben

- andere Teile erforderten aber signifikante manuelle Eingriffe
 - insbesondere die Markierung von Anweisungen war problematisch

Formalität konnte nicht vollständig durchgehalten werden

- insbesondere war die Abbildung von Ganzzahlen nicht 100%-ig korrekt
 - alle Ganzzahlen wurden auf denselben MALPAS IL Ganzzahltyp abgebildet
 - unabhängig von der Bitbreite (8-,16- oder 32-Bit) der Ganzzahl
 - falls nötig, wurde diese Unterscheidung manuell eingebracht



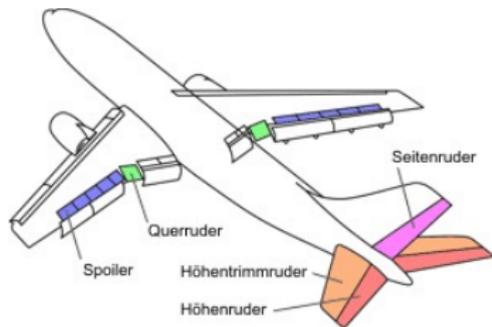
- 1 Überblick
- 2 Sizewell B
 - Überblick
 - Reaktorschutzsystem
 - Softwareverifikation
- 3 Airbus: Fly-by-Wire Flight Controls
 - Flugsteuerung
 - Flugsteuerung A320/A330/A340
 - Architekturoptimierung
 - Softwareverifikation@AIRBUS
- 4 Zusammenfassung



Flugsteuerung (engl. *Flight Control*)

Die Lenkung eines Flugzeugs.

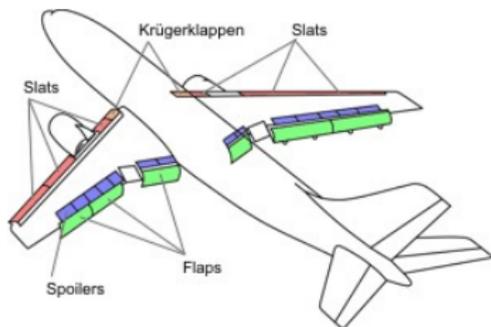
- umfasst **Steuerflächen** (engl. *control surfaces*) und ihre Ansteuerung
↪ Elemente um die Lage des Flugzeugs im Raum zu ändern
- **primäre Steuerflächen** eines Flugzeugs:



- **Querruder** (engl. *ailerons*)
↪ Rollen um die Längsachse
- **Spoiler** verringern den Auftrieb
↪ Landung, Querruder unterstützen
- **Seitenruder** (engl. *rudder*)
↪ Gieren um die Hochachse
- **Höhenruder** (engl. *elevator*)
↪ Kippen um die Querachse
- **Höhentrimmruder** (engl. *trimmable horizontal stabilizer, (THS)*)
↪ Höhensteuerung und Höhentrimmung



■ sekundäre Steuerflächen eines Flugzeugs:



- Vorflügel (engl. *slats*)
 - ~> höherer Auftrieb durch mehr Wölbung
- Krügerklappen (engl. *Kruger slats*)
 - ~> ähnlich dem Vorflügel, aber einfacher
- Landeklappen (engl. *flaps*)
 - ~> Auftriebshilfen für den Landeanflug
 - ~> ermöglichen verminderte Geschwindigkeit beim Landen

■ es gibt eine Reihe weitere sekundäre Steuerflächen

- Kippnasen, Spalt, Fowler, Spreiz-Klappen, Junkers-Doppelflügel, ...

■ Flugzeuge haben i. d. R. nur eine Auswahl sekundärer Steuerflächen

- ein Airbus A310 hat z. B. Vorflügel, Krügerklappen und Spaltklappen



- **mechanische Systeme** \leadsto direkte Verbindung zur Steuerfläche
 - ein „Lenkgestänge“ verbindet Steuerknüppel und Steuerflächen direkt
 - am Steuerknüppel ist der Luftwiderstand an der Steuerfläche spürbar
 - \leadsto problematisch bei großen Flugzeugen/hohen Geschwindigkeiten
- **elektrische/hydraulische Systeme** \leadsto motorisierte Aktoren
 - der Pilot steuert nur noch die Aktoren an, nicht mehr die Steuerfläche
 - Lenkbefehle öffnen Ventile oder setzen Servomotoren in Gang
 - \leadsto „Force Feedback“ wird notwendig, um dem Piloten Rückmeldung zu geben
 - **mechanische Ansteuerung** \leadsto „Lenkgestänge“ zum Aktor
 - **elektrische/elektronische Ansteuerung** \leadsto „Fly-by-Wire“
 - **analog** \leadsto Steuerbefehle werden direkt auf elektrische Signale umgesetzt
 - **digital** \leadsto computergesteuerte Umsetzung der Steuerbefehle
 - digitale „Fly-by-Wire“-Systeme haben diverse Vorteile
 - **Gewichtersparnis** im Vergleich zur Mechanik und zur Analogtechnik
 - Computerunterstützung: Autopilot, Erhöhung der Flugzeugstabilität, Absicherung des sicheren **Flugbereichs** (engl. *flight envelope*)



Historie: „Fly-by-Wire“

1930er Jahre „Tupolev ANT-20 Maxim Gorky“

- lange mechanische Steuerstrecken wurden elektrisch überbrückt

1958 „Avro Canada CF-105 Arrow“

- erstes militärische Serienflugzeug mit analogem „Fly-by-Wire“
- auch „Force Feedback“ (engl. *artificial feel*) wurde integriert

1969 „Concorde“

- erstes ziviles Verkehrsflugzeug mit analogem „Fly-by-Wire“

1972 „F-8 Crusader“

- Forschungsprojekt für digitales Fly-by-Wire“
 - eine digitale Flugsteuerung, drei analoge Backup-Systeme
- etwa zeitgleich: „Sukhoi T-4“ (UdSSR), „Hawker Hunter“ (GBR)

1977 „Space Shuttle Orbiter“ ~> komplett digitale Flugsteuerung

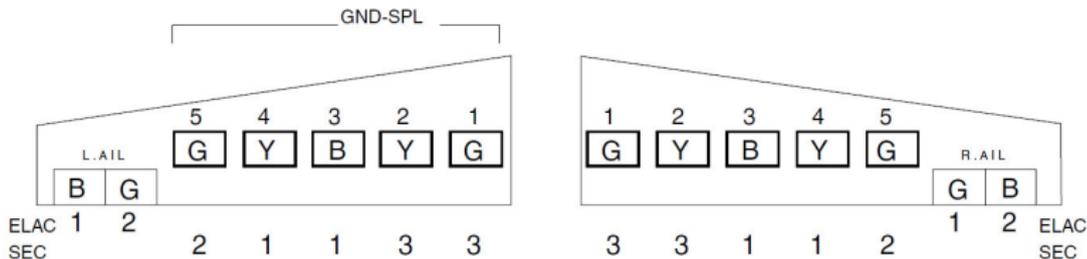
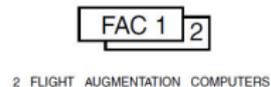
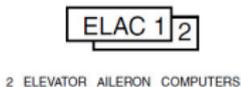
1984 „Airbus A320“

- erstes Verkehrsflugzeug mit vollständig digitaler Flugsteuerung
 - es existieren aber weiterhin elektrische Backup-Systeme

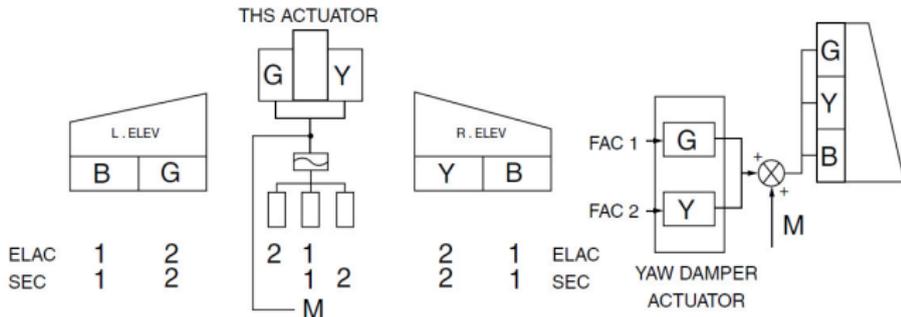


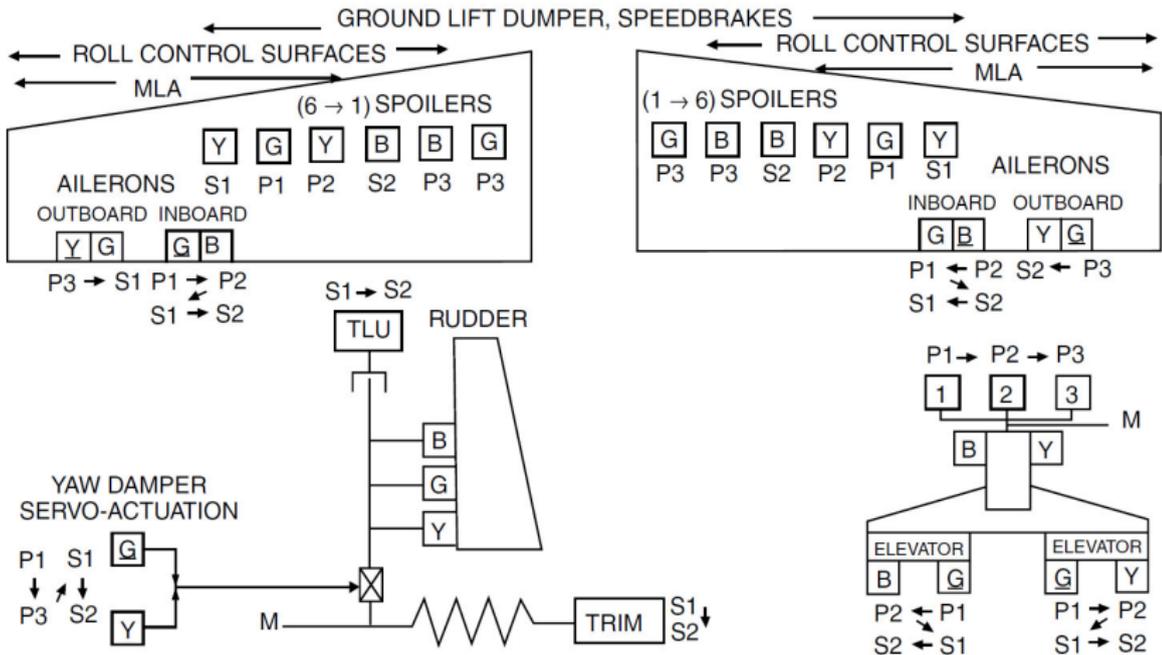
- **redundanter Aufbau** der Flugsteuerung
 - 7 (A320/321) bzw. 5 (A330/340) Computer zur Flugsteuerung
 - aktive Replikation und Implementierung von „fail-silent“-Verhalten
 - 3 Hydraulikkreisläufe zur Versorgung der Aktoren
 - > 2 getrennte Stromversorgungen durch Generatoren an den Triebwerken
 - Notstromversorgung (Windrad) falls alle Triebwerke ausfallen
- **diversitärer Aufbau** der Flugsteuerung
 - verschiedene Steuercomputer im A320/A321 [2]
 - 2 Elevator Aileron Computer (ELAC) ~ Motorola 68010 (Thomson-CSF)
 - 3 Spoiler Elevator Computer (SEC) ~ Intel 80186 (SFENA/Aerospatiale)
 - 2 Flight Augmentation Computer (FAC)
 - verschiedene Steuercomputer ab A330/A340 [7]
 - 3 Flight Control Primary Computers (PRIM) ~ Power PC
 - 2 Flight Control Secondary Computers (SEC) ~ Sharc
 - jeder Computer besteht aus einem **Kontroll-** und einem **Monitor-Kanal**
 - Verwendung unterschiedlicher Softwarepakete (z. B. Codegeneratoren)
 - ~ 4 Softwarepakete: 2 x ELAC, 2 x SEC im A320/321





HYDRAULIC
B Blue system
G Green system
Y Yellow system





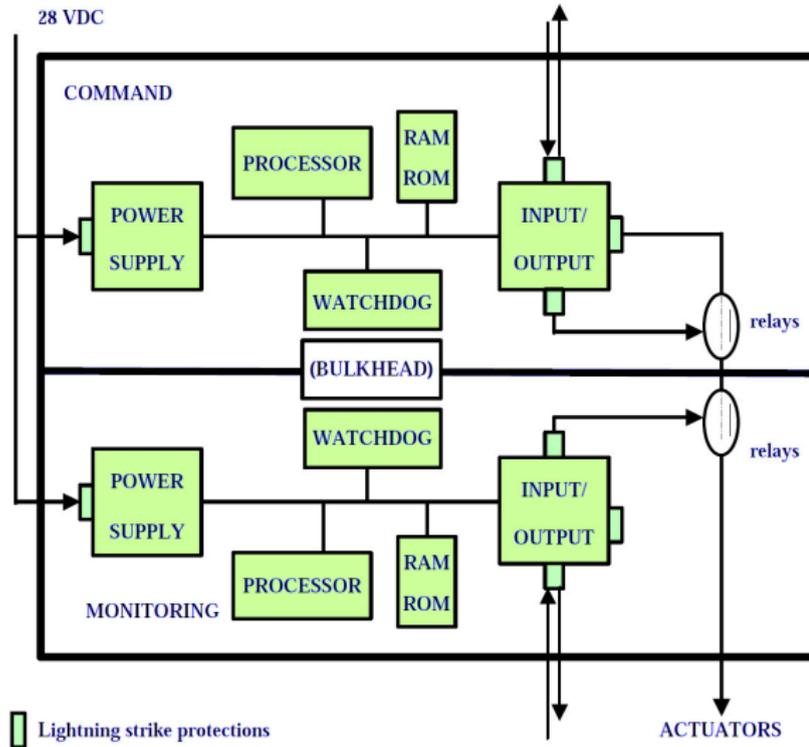
P : Flight controls primary computer
 S : Flight controls secondary computer

B, G, Y : Blue, Green, Yellow hydraulic circuits



Aufbau eines Steuercomputers der Flugsteuerung

Quelle Grafik: [7]

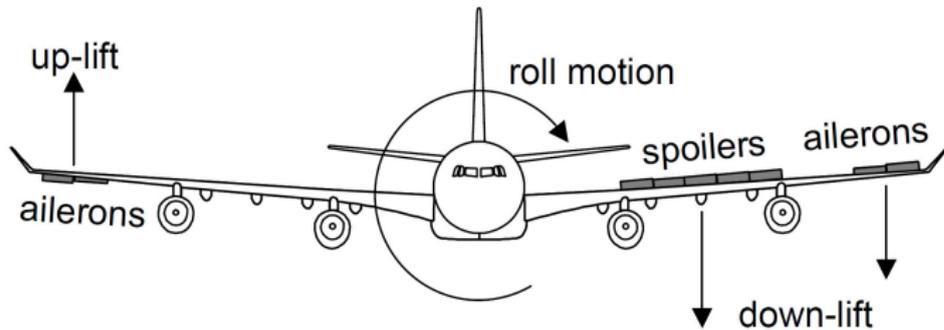


- **Selbstüberwachung** einzelner Steuercomputer
 - **stetige gegenseitige Überwachung** von Kontroll- und Monitor-Kanal
 - Vergleich von Ausgaben und ausgewählten Zustandsvariablen
 - übersteigt die Differenz einen gewissen Schwellwert \leadsto Fehler
 - **Selbsttests** werden zumindest **bei Systemstart** durchgeführt
 - Flugzeuge werden anders als Kernkraftwerke öfters neu gestartet
 - **Rekonfiguration** \leadsto „Graceful Degradation“
 - **Regelkreise zur Fluglageregelung** sind auf **Sensoren** angewiesen
 - andernfalls kann der Zustand des Flugzeugs nicht mehr erfasst werden
 - im Fokus: „**air data and inertial reference units**“ (ADIRUs)
 - Nicken \leadsto Unterstützung durch Beschleunigungssensoren und Gyroskope
 - \leadsto **funktionale Redundanz**
- \leadsto „Graceful Degradation“ \leadsto Rückzug auf eine direkte Flugsteuerung
- z. B. wenn alle ADIRUs ausgefallen sind
 - \leadsto keine computergestützte Fluglageregelung, kein gesicherter Flugbereich



- **Status Quo:** Redundanz und Diversität
 - Steuercomputer, Stromversorgungen, Antriebsstränge, ...
- **aber:** Wie nutzt man das am besten aus?
 - Wieviele Aktoren hängen an den einzelnen Steuerflächen?
 - Aus welchen Antriebssträngen werden diese Aktoren gespeist?
 - Welche Steuercomputer kontrollieren diese Aktoren?
- **Herausforderung:** Optimierte diese Zuordnung
 - gewährleiste dabei die Einhaltung einer minimalen Funktionalität
 - die Ausfallwahrscheinlichkeit muss über einer bestimmten Schwelle liegen
 - und erziele ein möglichst geringes Gewicht
 - **bisher:** manuelle ad-hoc Herangehensweise
 - Entscheidungen basieren vor allem auf Expertenwissen
 - Optimierung der Architektur durch „Trial & Error“





■ beteiligte Steuerflächen: Querruder und Spoiler

☞ eine **Fehlfunktion der Steuerflächen** verringert die **Rollrate**

- **verbleibende Rollrate** im Anbetracht des Fehlers f sei $p_\infty(f)$
- die **geforderte Rollrate** p_r hängt von der Fehlerwahrscheinlichkeit $\lambda(f)$ ab
 - häufige Fehler dürfen die Rollrate nur wenig beeinflussen

↪ **ausreichende Funktion** falls $m = \min_f p_\infty(f) / p_r(\lambda(f)) \geq 1$

- die geforderte Rollrate p_r wird immer erreicht, auch im Fehlerfall



Modellierung des Gewichts

- Gewicht $w(A)$ einer Architektur/Zuordnung A wird beeinflusst von
 - verwendete Aktoren, Verkabelung/-drahtung, Energieverbrauch
 - in frühen Entwicklungsstadien sind keine konkreten Werte verfügbar
- ↪ Modell für die Abschätzung des Gewichts ist notwendig
- **Gewichtsunterschied** zwischen Architektur A und einer Referenz R :

$$\delta w(A) = w(A) - w(R) = w(a_1, \dots, a_n) - w(r_1, \dots, r_n)$$

- Architektur A besteht dabei aus mehreren **Architekturentscheidungen** a_i
 - Welcher Typ Aktor wird verwendet? Welcher Computer steuert ihn? ...
- der **Gewichtseinfluss einer Entscheidung** a_i wird bestimmt durch

$$\delta_i w(a_i) = w(r_1, \dots, r_{i-1}, a_i, r_{i+1}, \dots, r_n) - w(R)$$

- summiert man diese Einflüsse auf, erhält man den Einfluss von A

$$\delta w(A) = \sum_{i=1}^n \delta_i w(a_i)$$

↪ Abschätzung des Gewichtseinflusses einer kompletten Architektur



- **Minimiere den Gewichtseinfluss** $\delta w(A)$ der Architektur A
 - gewährleiste dabei die **geforderte Rollrate**: $m(A) \geq 1$
 - und beachte folgende **technische Nebenbedingungen**:
 - geeignete Energieversorgung für jeden Aktor (Elektrik/Hydraulik)
 - jeder Aktor muss an mind. einen Computer angeschlossen sein
 - die Daten- und Stromleitungen für jeden Aktor liegen auf derselben Route
 - ein Spoiler wird von einem Computer gesteuert
 - ein Querruder wird von zwei Aktoren bewegt
 - ein Querruder wird von mind. einem elektronischem Antrieb gesteuert, es werden keine hybriden Antriebe verwendet
 - Aktoren an den Querrudern haben verschiedene Architekturen (also verschiedene Energieversorgung oder Steuercomputer)
 - Aktoren an den Querrudern sind mit einer identischen Anzahl von Steuercomputer verbunden



Kombinatorische Komplexität

- für jeden Aktor gibt es N_{act} Konfigurationsmöglichkeiten

$$N_{act} = \underbrace{(n_h + n_e + n_h n_e)}_{\text{Energie}} \cdot \underbrace{(n_c + n_c(n_c - 1))}_{\text{Steuerrechner}}$$

- $N_s = N_{act}$ Möglichkeiten für einen Spoiler, $N_a \approx N_{act}^2$ für ein Querruder
- $N = N_a^{n_a} N_s^{n_s}$ Möglichkeiten für n_a Querruder und n_s Spoiler



für reale Flugzeuge bedeutet das

		A320	A340 _{3H}	A340 _{2H2E}	A380
hydraulische Kreisläufe	n_h	3	3	2	2
elektrische Kreisläufe	n_h	0	0	2	2
Steuercomputer	n_c	5	5	6	6
Spoiler	n_s	8	10	10	12
Querruder	n_a	2	4	4	6
Kombinationen	N_{act}	75	75	288	288
Kombinationen Spoiler	N_s	75	75	288	288
Kombinationen Querruder	N_a	> 5000	> 5000	> 80000	> 80000
mögliche Architekturen	N	> 10^{22}	> 10^{33}	> 10^{44}	> 10^{59}



Reduktion der Kombinatorischen Komplexität

- Ausnutzung von Symmetrie bei den Spoilern $\leadsto n_s^n = n_s/2$
 - es wird immer nur ein Seite der Spoiler bewegt
- \leadsto bei den Querrudern ist dies nicht möglich
- Ausnutzung der technischen Randbedingungen
- \leadsto vorab werden infrage kommende Architekturen bestimmt
 - für Querruder \leadsto APA, für Spoiler \leadsto SPA



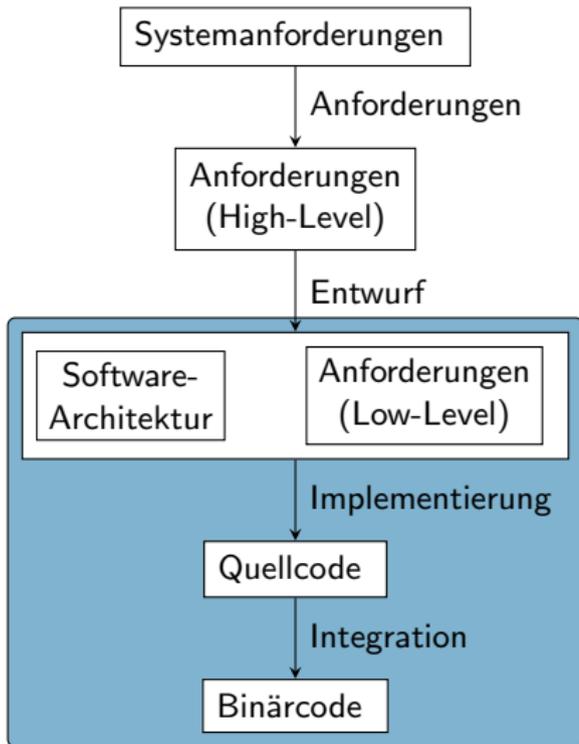
signifikante Reduktion des Suchraums:

		A320	A340 _{3H}	A340 _{2H2E}	A380
Spoiler	n_s^n	4	5	5	6
Querruder	n_a	2	4	4	6
SPA	N_s	21	21	36	36
APA	N_a	54	54	70	70
mögliche Architekturen	N	$< 10^9$	$< 10^{14}$	$< 10^{16}$	$< 10^{21}$

- Lösung durch einen **Branch&Bound-Algorithmus**
 - 3 hydraulische Kreisläufe \leadsto 3,1 kg Gewichtersparnis
 - 2 hydraulische/2 elektrische Kreisläufe \leadsto nur 1% Mehrgewicht
 - im Vgl. zu einem System, dass die geforderte Rollrate nicht erreicht



- alle Erzeugnisse des Entwicklungsprozess müssen verifiziert werden



- Einsatz von formalen Methoden auf den tieferen Abstraktionsebenen
- geeignete/geforderte Methoden sind:
 - deduktive Methoden
 - Caveat
 - Frama-C
 - abstrakte Interpretation
 - Astrée
 - Absint aiT
 - Absint Stackanalyser
 - Fluctuat
 - „certified compilation“
 - „translation validation“
 - verifizierter Übersetzer



- **Unit Proof** anstelle von **Unit Tests** \leadsto Caveat
 - Beschreibung von Anforderungen (Low-Level) mit der Caveat-Sprache
 - Spezifikation der Funktion mit Caveat in der Entwurfsphase
 - \leadsto jede C-Funktion wird mit Caveat gegen ihre Spezifikation geprüft
 - Anwendung bis dato in drei Projekten
 - Anwendung von Caveat erfordert eine dreitägige Schulung
- **Berechnung der WCET** \leadsto aiT
 - Anwendungsstruktur unterstützt die statische Bestimmung von WCETs
 - statisch berechnete Ablaufpläne, nicht-verdrängbare Ausführung
 - Einsatz bis dato in sechs Projekten (mehr werden folgen)
 - Anwendung durch die Softwareentwickler selbst
 - kein spezielles Training erforderlich
 - eine Ansprechperson für die Optimierung der Analyse
- **Berechnung des Stackbedarfs** \leadsto Stackanalyser
 - Ausschluss von Überläufen zur Laufzeit, Optimierung des Stackverbrauchs
 - Anwendung bei allen Projekten für eingebettete Systeme
 - derzeit zehn Projekte für A380 und A400M



- **Integration Proof** \rightsquigarrow mehr als **Unit Proof**
 - **Unit Proof** bezieht sich nur auf die aktuelle Prozedur
 - **Integration Proof** auf einen kompletten Teilbaum des Aufrufgraphen

- **Absenz von Laufzeitfehlern** \rightsquigarrow Astrée
 - für aus SCADE-Modellen generierter Code funktioniert das schon
 - der Automatisierungsgrad ist hier auch schon sehr hoch

- **Fehlerabschätzung von Fließkommaberechnungen** \rightsquigarrow Fluctuat
 - bisher: manuelle Analyse von Rundungsfehlern und Fehlerfortpflanzung \rightsquigarrow Fluctuat erlaubt dies bereits heute präzise und automatisch



- 1 Sicherung der **Ausführbarkeit** der Implementierung
 - der Quellcode darf kein undefiniertes Verhalten hervorrufen
 - ↪ unbedingte Einhaltung von
 - ISO/ANSI C Standard, IEEE 754 Floating-Point-Standard
 - Kodierungs- und Codegenerierungsrichtlinien
 - Zeitbeschränkungen
 - Anforderungen an die numerische Präzision
 - Synchronisation und Kommunikation
 - ↪ ihre Verletzung kompromittiert benutzerdefinierte Eigenschaften
- 2 Sicherung **benutzerdefinierter Eigenschaften**
 - durch **Unit Proof** und/oder **Integration Proof**
- 3 „**Certified Compilation**“
 - der Quellcode muss auch korrekt auf die Hardware übertragen werden



- der Einsatz formaler Methoden ist nicht nur „guter Wille“
 - solche Werkzeuge und Anstrengungen müssen sich auch bezahlt machen



Airbus stellt folgende Anforderungen

- **Korrektheit** der Methode
 - man muss sich auf die Ergebnisse verlassen können
- **direkte Anwendbarkeit** auf den Quellcode der Anwendung
 - keine Entwicklung zusätzlicher Modelle, ...
- **Benutzbarkeit** durch normale Softwareentwickler
 - die Werkzeuge erfordern keine Gurus für formale Methoden
 - sie benötigen auch keine Super-Computer für ihre Ausführung
- **Optimierung** des Entwicklungsprozesses
 - formale Methoden müssen besser sein als bestehende Techniken
- Verbesserung der **Zertifizierbarkeit**
 - formale Methoden müssen die Zertifizierung erleichtern



- 1 Überblick
- 2 Sizewell B
 - Überblick
 - Reaktorschutzsystem
 - Softwareverifikation
- 3 Airbus: Fly-by-Wire Flight Controls
 - Flugsteuerung
 - Flugsteuerung A320/A330/A340
 - Architekturoptimierung
 - Softwareverifikation@AIRBUS
- 4 Zusammenfassung



Sizewell B \leadsto primäres Reaktorschutzsystem

- einziger Zweck: sichere Abschaltung des Reaktors

Airbus \leadsto digitale „Fly-by-Wire“-Flugsteuerung

- die Lenkung moderner Verkehrsflugzeuge

Redundanz \leadsto Absicherung gegen Systemausfälle

- bis 7-fach redundante Systeme

Diversität \leadsto Abfedern von Software-Defekten

- unterschiedliche Hardware und Software

Isolation \leadsto Abschottung der einzelnen Replikate

- technisch \mapsto optische Kommunikationsmedien
- zeitlich \mapsto nicht-gekoppelte, eigenständige Rechner
- räumlich \mapsto verschiedene Aufstellorte und Kabelrouten

Verifikation \leadsto umfangreiche statische Prüfung von Software

- vielschichtiger Prozess, Betrachtung von Quell- und Binärcode



- [1] BAUER, C. ; LAGADEC, K. ; BÈS, C. ; MONGEAU, M. :
Flight Control System Architecture Optimization for Fly-By-Wire Airliners.
In: *Journal of Guidance, Control, and Dynamics* 30 (2007), Jul., Nr. 4, S. 1023–1029.
<http://dx.doi.org/10.2514/1.26311>. –
DOI 10.2514/1.26311
- [2] *Kapitel 12.*
In: BRIERE, D. ; FAVRE, C. ; TRAVERSE, P. :
Electrical Flight Controls, From Airbus A320/A330/A340 to Future Military Transport Aircraft: A Family of Fault-Tolerant Systems.
CRC Press LLC, 2001 (The Electrical Engineering Handbook Series). –
ISBN 978-0849383489
- [3] BUTTLE, D. L.:
Verification of Compiled Code.
Eindhoven, The Netherlands, University of York, Diss., Jan. 2001. –
262 S.
- [4] MOUTREY, G. ; REMLEY, G. :
Sizewell B power station primary protection system design application overview.
In: *International Conference on Electrical and Control Aspects of the Sizewell B PWR*, 1992. –
ISBN 0-85295-550-8, S. 221–231



- [5] PAVEY, D. J. ; WINSBORROW, L. A.:
Demonstrating Equivalence of Source Code and PROM Contents.
In: *The Computer Journal* 36 (1993), Apr., Nr. 7, S. 654–667.
<http://dx.doi.org/10.1093/comjnl/36.7.654>. –
DOI 10.1093/comjnl/36.7.654
- [6] SOUYRIS, J. ; WIELS, V. ; DELMAS, D. ; DELSENY, H. :
Formal Verification of Avionics Software Products.
In: *Proceedings of the 2nd World Congress on Formal Methods (FM '09)*.
Heidelberg, Germany : Springer-Verlag, 2009. –
ISBN 978-3-642-05088-6, S. 532–546
- [7] TRAVERSE, P. ; LACAZE, I. ; SOUYRIS, J. :
Airbus Fly-By-Wire: A Process Toward Total Dependability.
In: *Proceedings of the 25th Congress of International Council of the Aeronautical Sciences (ICAS '06)*, 2006



- [8] WARD, N. J.:
The Rigorous Retrospective Static Analysis of the Sizewell 'B' Primary Protection System Software.
In: GÓRSKI, J. (Hrsg.): *Proceedings of the 12th International Conference on Computer Safety, Reliability, and Security (SAFECOMP '93)*.
Heidelberg, Germany : Springer-Verlag, Okt. 1993. –
ISBN 3-540-19838-5, S. 171-181

