AUTOCODING SICHERHEITSRELEVANTER SOFTWARE IN DER LUFTFAHRT

Dieter Reiners, DaimlerChrysler Forschung und Technologie Frankfurt
Ines Fey, DaimlerChrysler Forschung und Technologie Berlin
Dr.Carsten Thomas, DaimlerChrysler Aerospace Airbus Hamburg
et.al.

Kurzfassung: *Softwareentwicklung ist ein wesentlicher Kostenfaktor im Entstehungsprozeß neuer Flugzeuge. Die Softwareentwicklung muß besser, schneller und billiger werden. Im Projekt " Qualified System Development Process" hat die DaimlerChrysler Forschung in Frankfurt und Berlin in Kooperation mit DaimlerChrysler Aerospace (Dasa) Airbus ein methodisches Konzept für einen RTCA/DO-178B-konformen, durchgängigen Prozeß zur Entwicklung sicherheitsrelevanter Software entwickelt. Dieser geht aus von einer Spezifikation, die in einer leistungsfähigen Systemsimulation ablauffähig ist. Aus dem Simulationsmodell wird automatisch der zertifizierbare Code für die sicherheitsrelevanten Steuer- und Regelsysteme erzeugt. Im Projekt QSDP wurden die speziellen Anforderungen der Luftfahrtindustrie und die organisatorischen und kommerziellen Randbedingungen bei Dasa Airbus und deren Partnern analysiert. Die bei Dasa Airbus verwendeten Software-Entwicklungstools wurden erfaßt, verschiedene Szenarien von Entwicklungsprozessen erarbeitet und die Werkzeugumgebung für ein zusammen mit Dasa Airbus ausgewähltes Szenario prototypisch realisiert. Dabei wird in einer Matlab-Umgebung das Control-System, der zu steuernde technische Prozeß und seine Umgebungsbedingungen ganzheitlich modelliert. Die Realzeitsimulation des Prozesses und seiner Umgebungsbedingungen sind nicht zulassungsrelevant und können daher mit den verfügbaren Codegeneratoren codiert werden. Der als Control-Software in die entsprechenden Controller zu ladende Code darf dagegen nur mit einem qualifizierten Codegenerator automatisch erzeugt werden. Der beschriebene Prozeß wurde an einem Anwendungsbeispiel aus dem Verantwortungsbereich von Dasa Airbus evaluiert.*

Das Projekt wurde durchgeführt von der DaimlerChrysler Forschung in Berlin und Frankfurt und mit Mitteln des Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie unter der Nummer 20F9701A gefördert.

1 EINLEITUNG

Software-basierte Systeme übernehmen zunehmend sicherheitsrelevante Aufgaben in den Produkten der zivilen Luftfahrt. Flugsteuerungsfunktionen werden heute zu einem großen Teil in Form von Software realisiert. Die Anwendung der fly–by–wire Technologie setzt sich zunehmend durch. [2.1]. Zur Gewährleistung der notwendigen Sicherheit gelten hohe Qualitätsanforderungen, was sich u.a. in besonderen Vorschriften und Standards für den Entwicklungsprozeß niederschlägt. Gleichzeitig wachsen Funktionsumfang und Komplexität dieser Systeme und damit auch die Herausforderungen in Zusammenhang mit der Einhaltung von Entwicklungszeiten und Kostenrahmen.

Um einen effizienten Ablauf der Software-Entwicklung zu gewährleisten und gleichzeitig die Sicherheitsaspekte ausreichend zu berücksichtigen, müssen die Entwicklungsprozesse optimiert werden. Sie müssen "schneller, besser und billiger" werden. Methoden und Werkzeuge sind zu einem durchgängigen Entwicklungsprozeß zusammenzuführen.

Im Projekt " Qualified System Development Process" (QSDP) wurden die prozeßspezifischen, kommerziellen und organisatorischen Anforderungen an einen durchgängigen Entwicklungsprozeß analysiert. Es wurden verfügbare SW-Werkzeuge untersucht und bewertet, und es wurden Szenarien für eine optimierte QSDP-Entwicklungsumgebung ausgearbeitet. Aus der Bewertung dieser Szenarien wurde ein methodisches Konzept eines "Qualified System Development Process" abgeleitet. Die Elemente dieses methodischen Konzeptes wurden spezifiziert. Abschließend wurde eine prototypische Realisierung des Entwicklungsprozesses an einem flugzeugrelevanten Beispiel aus dem Verantwortungsbereich von Dasa Airbus evaluiert und so die Machbarkeit eines qualifizierbaren Prozesses zur automatisierten Entwicklung sicher-heitsrelevanter Software demonstriert. Wie Bild 1 zeigt, umfaßt der QSDP die Validierung der ablauffähigen Systemspezifikation in einer leistungs-fähigen Simulationsumgebung, die Entwicklung der spezifizierten Steuer- und Regelungsfunktionen sowie die automatische Generierung des Zielcodes für den Steuerungsrechner.

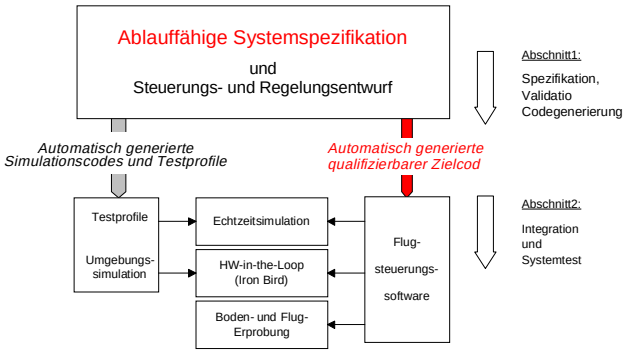

Bild 1: Allgemeine QSDP-Zielvorstellung

Das QSDP-Projekt wurde durchgeführt von der DaimlerChrysler Forschung in Berlin und Frankfurt und mit Mitteln des Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie unter der Nummer 20F9701A gefördert.

2 Ziel des QSDP-Projektes

Der Entwicklungsprozeß kann entsprechend der vereinfachten V-Modell-Darstellung in BILD 2 in die Phasen Systemspezifikation, Komponentenspezifikation, Hardware- / Softwaredesign, Realisierung bzw. Implementierung, Hardware-/Softwareintegration, Komponentenintegration und Systemintegration unterteilt werden. Die konstruktiven Phasen sind auf der linken Seite des V und die verschiedenen Integrationsschritte auf der rechten Seite des V dargestellt. Die Querverbindungen zwischen den konstruktiven und den integrativen Schritten sind verschiedene Tests vom HW-/SW-Integrationstest bis zum Systemtest.

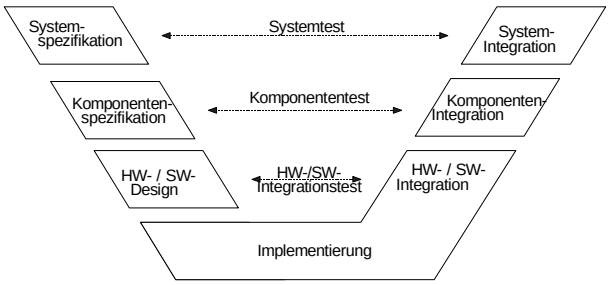

Bild 2: Entwicklungsphasen im vereinfachten V-Modell

In der Phase der Systemspezifikation werden die Anforderungen an ein zu entwickelndes System spezifiziert. Anhand der ermittelten Anforderungen werden die Systemarchitektur und die Anforderungen an die Komponenten festgelegt. Für jede Komponente läuft der weitere Entwicklungsprozeß zunächst separat weiter. Hard- und Software-Design bestehen demnach aus einer Reihe von parallelen Entwicklungsprozessen. Aus den Anforderungen an die Komponenten wird die Partitionierung in Hard- und Softwarekomponenten abgeleitet. Danach erfolgen Hard- und Software-Entwicklung der Komponenten getrennt. Im nächsten Schritt wird die Hardware gebaut und der aus dem Quell-Code generierte Objektcode in die Hardware implementiert.

Die Eigenschaften der realen Komponente werden im Komponententest mit der Spezifikation der Komponente verglichen. Abschließend werden alle Komponenten zum realen System zusammengebaut bzw. integriert, und das Verhalten des realen Systems wird gegen die Systemspezifikation geprüft. Erkannte Fehler werden in wiederholten Bearbeitungszyklen beseitigt. Je später ein Fehler erkannt wird, um so aufwendiger sind im Allgemeinen die erforderlichen Korrekturmaßnahmen. Aus diesem Grund ist es eine wirtschaftliche Notwendigkeit, bereits möglichst früh in den konstruktiven Phasen entsprechende Verifikationen bzw. Validierungen der Spezifikationen durchzuführen. Dies soll an den beiden folgenden Darstellungen näher erläutert werden.

Auch der Entwicklungszyklus nach Bild 3 folgt prinzipiell dem V-Modell. Er beginnt mit der Spezifikation. Es folgen die Modellierung, die Codierung der Software, die Implementierung des Codes in die Komponente und die entsprechenden Integrationsschritte und abschließend die Validierung des realen Systems gegen die Systemspezifikation. Hier ist leicht zu erkennen, daß Fehler in der Spezifikation erst bei der abschließenden Validierung erkannt werden. Zur Korrektur des Fehlers wäre der komplette Zyklus erneut zu durchlaufen.

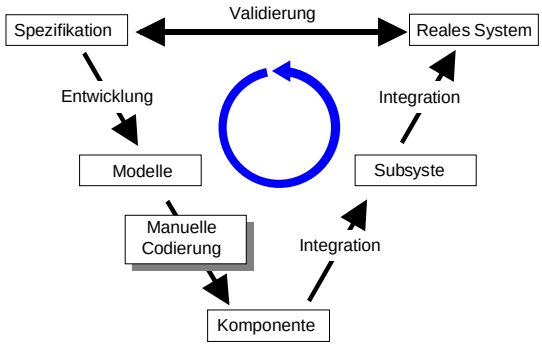

Bild 3: Früherer Software-Entwicklungsprozeß

Das Durchlaufen dieser großen Schleife ist mit entsprechend großen Entwicklungszeiten und Kosten verbunden. Dies gilt um so mehr, als bisher die Codierung der Software praktisch manuell erfolgen muß. Der Qualified System Development Process setzt daher, wie in Bild 4 gezeigt, an zwei Stellen des Entwicklungsprozesses an:

Er versucht mit der ablauffähigen Spezifikation bereits die Spezifikation der Anforderungen in einer Simulationsumgebung zu validieren. Dieser erste QSDP-Gedanke wird durch den kleinen Kreis dargestellt. Dieser Kreis – die simulationsbasierte Validierung der Systemspezifikation - wird solange durchlaufen, bis in der Simulation keine Fehler mehr erkannt werden.

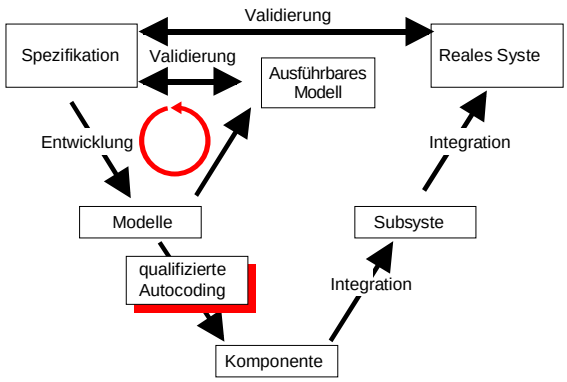

Bild 4: Qualified System Development Process

Der zweite QSDP-Gedanke ist es nun, den Zielcode der Komponenten mit einem qualifizierten, automatischen Codegenerator weitgehend automatisch zu erzeugen, so daß die durchzuführenden SW-Tests deutlich reduziert werden oder entfallen können. Da der Testaufwand für manuell erzeugten Code einen erheblichen Teil des Entwicklungsaufwandes ausmacht, wird mit der qualifizierten Automatisierung der SW-Codierung eine deutliche Reduzierung der Entwicklungszeiten und Kosten realisiert. Aufgabe des QSDP-Projektes war es deshalb, die Machbarkeit eines solchen

qualifizierten Software-Entwicklungsprozesses mit simulationsbasierter Validierung der Spezifikation und automatischer Code-Generierung zu demonstrieren.

3. Analyse der Anforderungen

Die Analyse der Anforderungen hat gezeigt, daß QSDP konform zur RTCA/DO-178 sein muß. Außerdem müssen die kommerziellen und organisatorischen Randbedingungen bei der Realisierung und der Einführung des neuen Prozesses deutliche wirtschaftliche Vorteile erkennen lassen. Im Folgenden soll deshalb besonders auf die RTCA/DO-178 eingegangen werden. Anschließend werden die kommerziellen und wirtschaftlichen Rahmenbedingungen kurz vorgestellt. Von besonderem Einfluß ist hierbei auch die Frage der Verwendbarkeit der bisher genutzten Software-Entwicklungstools.

3.1 Softwareentwicklung nach RTCA/DO-178

Bei der Software-Entwicklung im Umfeld der Arbeiten von Dasa Airbus sind die Vorschriften JAR25 [1.3], ABD0100 [1.2] und RTCA/DO-178 [1.1] zu berücksichtigen. Von besonderer Bedeutung für die Zulassung eines Qualified System Development Process ist dabei seine Konformität zur RTCA/DO-178.

Der Standard RTCA/DO-178 legt die Zertifizierungsaspekte für fliegende Systeme und Ausrüstungen im Hinblick auf die verwendete Software dar. Schwerpunkt ist die Festlegung von Zielen und Ergebnissen, die während des Entwicklungsprozesses zu erarbeiten sind. Methoden und Werkzeuge, die das Erreichen dieser Zielsetzung bewirken, werden nicht festgelegt. Der Standard beschreibt sämtliche Tätigkeiten der Software-Entwicklung. Diese sind den verschiedenen Prozessen zugeordnet, in die der Gesamt-Entwicklungsprozeß unterteilt wird. DO-178 betrachtet die Entwicklung von Avionik-Software als eine Reihe *von Software Life Cycle Processes*, welche als Teil des *System Life Cycle Processes*, anzusehen sind Die Beziehung zwischen System und Software Life Cycle Processes, der Einfluß der *System Requirements* auf die zu entwickelnde Software und umgekehrt die Wirkung der Software auf das Verhalten des Systems werden erläutert. Für jeden Prozeß benennt DO-178 die zu erfüllenden Voraussetzungen und die Ziele bzw. Ergebnisse und gibt Empfehlungen, welche Aktivitäten der Zielerreichung und damit der Zertifizierung des Systems dienen können. Der Software-Entwicklungsprozeß nach DO-178 ist in Bild 5 dargestellt. DO-178 unterteilt diesen Prozeß in

- Planung (Software Planning Process)
- Begleitprozesse (Integral Processes)
- Software-Erstellung (SW Development Processes)

Software Planning Process

Dem eigentlichen Entwicklungsprozeß ist ein Planungsprozeß vorgelagert, in dessen Verlauf sämtliche für den Entwicklungsprozeß maßgebenden Pläne erstellt und die einzuhaltenden Standards - z.B. *Software Code Standards* - festgelegt werden. Als Ergebnis dieses Planungsprozesses liegt unter anderem ein Plan für das projektspezifische Zulassungsvorgehen im *Plan for Software Aspects of Certification* vor. Sämtliche in dieser Phase erstellten Dokumente werden unter dem *Software Configuration Management Process* verwaltet.

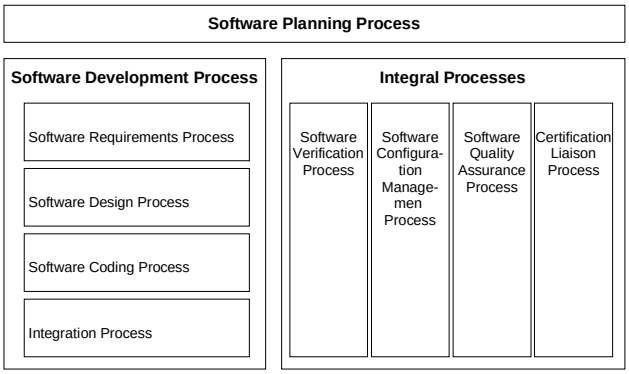

Bild 5: SW-Entwicklung nach RTCA/DO-178

Software Development Process

Der SW Development Process umfaßt lt. DO-178:

1. Software Requirements Process
2. Software Design Process
3. Software Coding Process
4. Integration Process

Im *Software Requirements Process* erfolgt die Ableitung von Anforderungen für die Software aus den Systemanforderungen. Außerdem werden zusätzliche Anforderungen an die Software ermittelt, die sich aus der Realisierung der Systemanforderungen in Software ergeben können.

Während des *Software Design Process* wird eine Verfeinerung der Software-Anforderungen durchgeführt. Dabei wird eine zur Umsetzung der Anforderungen geeignete Software-Architektur entwickelt, und die einzelnen Software-Komponenten werden festgelegt. Ergänzende Anforderungen, die sich aus der gewählten Architektur ergeben, sind ebenfalls an dieser Stelle zu definieren

Die Implementierung der Anforderungen in den Quell- bzw. Objekt-Code erfolgt während des *Software Coding Process*. Die neben dem Quell-Code notwendigen Daten zur Erzeugung von ausführbarem Code, wie beispielsweise Compiler-Anweisungen, werden gleichfalls in diesem Prozeß festgelegt.

Ziel des *Integration Process* ist die ausführbare, möglichst fehlerfreie auf die Ziel-Hardware geladene Software. Ergebnis ist das Software-Produkt, welches die Software Requirements Data, die Design Description, den Source-Code und den Object-Code umfaßt Für die im Verlauf des Software Development Process erzeugten Dokumente wird eine geordnete Verwaltung gefordert. Außerdem soll der Zugriff auf die Ergebnisse möglichst einfach sein.

Integral Processes

Die Erstellung der Software wird von weiteren Prozessen begleitet, die der Qualitätssicherung, der Dokumentation des Entwicklungsvorganges und vor allem dem Erlangen der Zulassung dienen.

Die Begleitprozesse setzen sich zusammen aus:

1. *Software Verification Process*
2. *Software Configuration Management Process*
3. *Software Quality Assurance Process*
4. *Certification Liaison Process*

Im Rahmen des *Software Verification Process* sollen Fehler in den erstellten Dokumenten aufgedeckt werden. Entsprechend ihrem unterschiedlichen Inhalt lassen sich diese Dokumente in den Kategorien Entwicklungsdokumente (z.B. Entwurfsbeschreibung, Verifikationsergebnisse), Erzeugnisse (Quell- und Objekt-Code), sowie begleitende Dokumente (SW-Entwicklungsplan und Verifikationsplan) unterteilen.

Die Verifikation bezieht sich somit nicht nur auf den Objekt-Code als letztendliches Ergebnis der SW-Erstellung. Die Prüfung erfolgt mit Hilfe von Reviews, Analysen und Tests. Reviews und Analysen sollen vorrangig die Konsistenz von Anforderungsanalyse, Entwurf und Codierung sowie die Einhaltung von Standards sicherstellen. Die Tests sollen zeigen, daß der Objekt-Code die Anforderungen erfüllt. Der Test erstreckt sich über die drei Ebenen Unit-Test, SW-Integrationstest sowie HW-/SW-Integrationstest. dem Test sind außerdem Analysen zugeordnet, wie die Messung der Strukturüberdeckung (Coverage Analysis) und die Identifizierung der berücksichtigten Anforderungen(*Requirements Coverage*).

Das *Software Configuration Management* Process soll die Konsistenz und die Verfügbarkeit der im Verlauf der Entwicklung entstandenen Dokumente sicherstellen. Außerdem ist in diesem Zusammenhang zu gewährleisten, daß Änderungen nachvollziehbar sind. Neben den Dokumenten bezieht das Konfigurations-management auch die verwendeten Werkzeuge ein.

Die Aufgabe des *Software Quality Assurance Process* besteht in erster Linie in der Überwachung der Entwicklungsprozesse. Zum Abschluß der SW-Entwicklung werden die Entwicklungsdokumente auf Vollständigkeit geprüft und die Archivierung des ausführbaren Objekt-Codes abgesichert.

Ziel des *Certification Liaison Process* ist es, Einvernehmen zwischen Entwicklern und Zulassungsbehörden herzustellen. Am Ende der Entwicklung belegt eine separates Dokument für die Zulassungsbehörden, daß die Vorgaben *des Plan for Software Aspects of Certification* eingehalten wurden.

Die Aktivitäten der *Integral Processes* dienen im Wesentlichen der Nachweisführung, daß die geplanten Ergebnisse der SW-Erstellung erreicht wurden. Im Rahmen der *Integral Processes* wird außerdem die Einhaltung der Pläne und der Standards wie beispielsweise der Implementierungsrichtlinien oder der Namenskonventionen überwacht. Die entsprechenden Begleitdokumente bzw. Nachweise sind unter dem Begriff *Software Life Cycle Date* zusammengefaßt.

Die genannten *Software Development Processes* können den verschiedenen Phasen der V-Modell-Darstellung des Entwicklungsprozesses zugeordnet werden. So gehören diese Prozesse mit Ausnahme der Integration zur linken Seite des V-Modells. Sie werden auch als konstruktive Aktivitäten bezeichnet. Der Integrationsprozeß bezieht sich auf die rechte Seite des V-Modells.

Der *Software Verifikation Process* entspricht den Verifikationsaktivitäten, die als Verbindungen zwischen den beiden Seiten des V-Modells eingezeichnet sind.

Die anderen *Integral Processes* berücksichtigt die V-Modell-Darstellung nicht.

3.2 Kommerzieller & organisatorischer Rahmen

Mit QSDP ist eine durchgängige Entwicklungs-umgebung geplant, mit deren Hilfe zertifizierbare Software für Steuer- und Regelsysteme in Flugzeugen weitgehend automatisch aus grafischen Spezifikationen generiert werden kann.

Die kommerzielle Realisierung eines solchen Prozesses erfordert eine enge Abstimmung zwischen DaimlerChrysler Forschung, Dasa Airbus und den europäischen Airbus-Partnern, den Airbus-Zulieferern und nicht zuletzt den kommerziellen Software-Herstellern. Diese müssen ihre verfügbaren Tools mit Blick auf den Qualified System Development Process überarbeiten, und ggfs. Brücken zu den Softwaretools anderer SW-Hersteller bauen. Die Airbus-Partner und ihre Zulieferer müssen gemeinsame Vorgehensweisen und Schnittstellen im Entwicklungs- und Fertigungsprozeß vereinbaren, um für die SW-Hersteller eine stabile wirtschaftliche Basis bieten zu können.

Dabei sind natürlich auch die bei den verschiedenen Partnern heute im Einsatz befindlichen Software-Entwicklungsumgebungen mit ihren Investitionen in Rechnerhardware, Software-Lizenzen und Ausbildung des Personals zu berücksichtigen.

3.3 Heute verwendete Software

Bei der Anforderungsanalyse wurden bei den Partnern u.a. folgende im Einsatz befindliche Tools identifiziert:

Spezifikationswerkzeuge
Matlab, MatrixX, SCADE, HOSTESS, Statemate, Teamwork

Simulatoren
Matlab, MatrixX

Codegeneratoren
SCADE, HOSTESS,

Prüf- und Testwerkzeuge
Logisscope

Dok-/Code-Management
RCS&CVS, SSCS, Clear Case

Diese Werkzeuge wurden zusammen mit einer Vielzahl weiterer am Markt verfügbarer Softwaretools auf ihre Eignung für QSDP analysiert. Dabei wurden sowohl die technischen, wie auch die kommerziellen und organisatorischen Rahmenbedingungen berücksichtigt.

4 Qualified System Development Process

4.1 Szenario

Ausgehend von den in der Analyse identifizierten Softwaretools wurden unter Berücksichtigung der Anforderungen der DO-178 und der organisatorischen und kommerziellen Rahmenbedingungen für den durchgängigen qualifizierbaren Entwicklungsprozeß verschiedene Szenarien für einen Qualified System Development Process entwickelt.

3 Typen von Szenarien werden unterschieden, deren Zahl sich durch diverse Ergänzungen vervielfältigt:

1. Halbautomatische Code-Generierung mit zwei Entwurfswerkzeugen und Modellkonverter
2. Automatische Code-Generierung mit einem Entwurfswerkzeug und Modellkonverter
3. Automatische Code-Generierung mit einem Entwurfswerkzeug ohne Modellkonverter

Kriterien für die Auswahl waren:

- Kosten für die Realisierung der Werkzeugkette
- Kosten für die Installation eines Arbeitsplatzes
- Schulungskosten
- Zeitlicher Rahmen
- Wahrscheinlichkeit der Umsetzung
- Einbettung in den Gesamtentwicklungsprozeß
- Flexibilität und Stabilität
- Nutzen der Werkzeugkette

Die besten Umsetzungserfolge wurden für ein Szenario identifiziert, dessen Struktur in Bild 6 auf der nächsten Seite dargestellt ist. Der vorgestellte Entwicklungsprozeß verbindet die Vorteile zweier leistungsfähiger kommerzieller Softwarepakete. Er erlaubt die Modellierung und Simulation mit den bei den Systementwicklern eingeführten, komfortablen Entwicklungs- und Simulationstools Matlab/Simulink oder MatrixX und nutzt zum Anderen den in vielen sicherheitsrelevanten Systemen besonders im Luftfahrtbereich erprobten Autocode-Generator des Programmpaketes SCADE. Bei dem ausgewählten Szenario wird die grafische Spezifikation mit Matlab durchgeführt. MatrixX ist gleichermaßen geeignet.

Das **Development und Simulation Tool** (DST) Matlab verfügt über eine Vielzahl von Toolboxen mit Modellierungshilfen, Stimulatoren und Analysewerkzeugen für effiziente Systemanalysen. Die mit diesen Werkzeugen mitgelieferten Autocode-Generatoren sind dagegen nicht für die RTCA/DO-178 konforme Entwicklung qualifiziert. Besonders kritisch sind in diesem Zusammenhang auch die komplexen Datenstrukturen und Rechenoperationen der Simulationswerkzeuge.

Die Code-Generierung erfolgt mit dem **SCADE-Code-Generator**. Dieser ist im Gegensatz zu den DST in mehreren Projekten qualifiziert. Er baut auf einfachen direkt adressierten Datenstrukturen und einer einfach strukturierten und damit leicht qualifizierbaren **Operator Library** auf, die in der SCADE-Spezifikationssprache LUSTRE geschrieben ist.

Der Entwickler arbeitet hauptsächlich in der ihm vertrauten Matlab/Simulink-Oberfläche.

Die Übersetzbarkeit von Matlab nach SCADE ist dadurch gesichert, daß bei der Funktionsspezifikation der in C-Code umzusetzenden Funktionen auch im Development and Simulation Tool nur SCADE-Operatoren verwendet werden. Diese Operatoren werden aus der SCADE Operator Library exportiert und in Matlab als externe DST-Library Operatoren bereitgestellt.

4.2 Entwicklungsablauf

Aufbauend auf dem beschriebenen Szenario ist der Entwicklungsablauf bei Verwendung des QSDP in zwei Aktionsstränge gegliedert. In Bild 6 von oben nach unten erfolgt zum einen die automatische Umsetzung des im **Development and Simulation Tool** (DST) entwickelten Specification Models in den **C-Code** für die Steuerungsrechner. Dieser Arbeitsgang ist ein grundlegender Bestandteil eines jeden Entwicklungsprojektes.

Die Spezifikation beschreibt die Anforderungen (**Requirements**), des zu entwickelnden Systems, enthält die Modelle (**Model**) des zu steuernden Prozesses mit all seinen Randbedingungen (**Constraints**) und Entwicklungs- und Optimierungsziele (**Design Goals**) für die Steuer- und Regelfunktionen. Die Steuerfunktionen werden in **Statecharts** abgebildet, die regelungstechnischen Zusammenhänge in **Blockdiagrams** beschrieben.

Prozeß und Steuerfunktionen werden ganzheitlich im **DST** modelliert und simuliert. Das gesamte Modell wird in einer **Model Specification im DST-Format** abgespeichert. Zusatzinformationen wie Modellparameter werden im **Data Dictionary** abgelegt.

Mit einem QSDP-spezifischen Translator wird dieser Datensatz zunächst auf Syntax-Fehler überprüft. Hierzu dient ein **DST-Model-Parser**. Das Ergebnis dieses Prüfvorganges ist eine **Model Specification**, die in einem **Intermediary Format** abgelegt wird.

Die darin enthaltenen Informationen werden unter Berücksichtigung des **Data Dictionaries** mit dem **To-SCADE-Translator** in ein SCADE-konformen Datensatz umgesetzt. Dieser Datensatz wird mit dem **SCADE-Checker** auf die Einhaltung der SCADE-Syntax und Sematik überprüft.

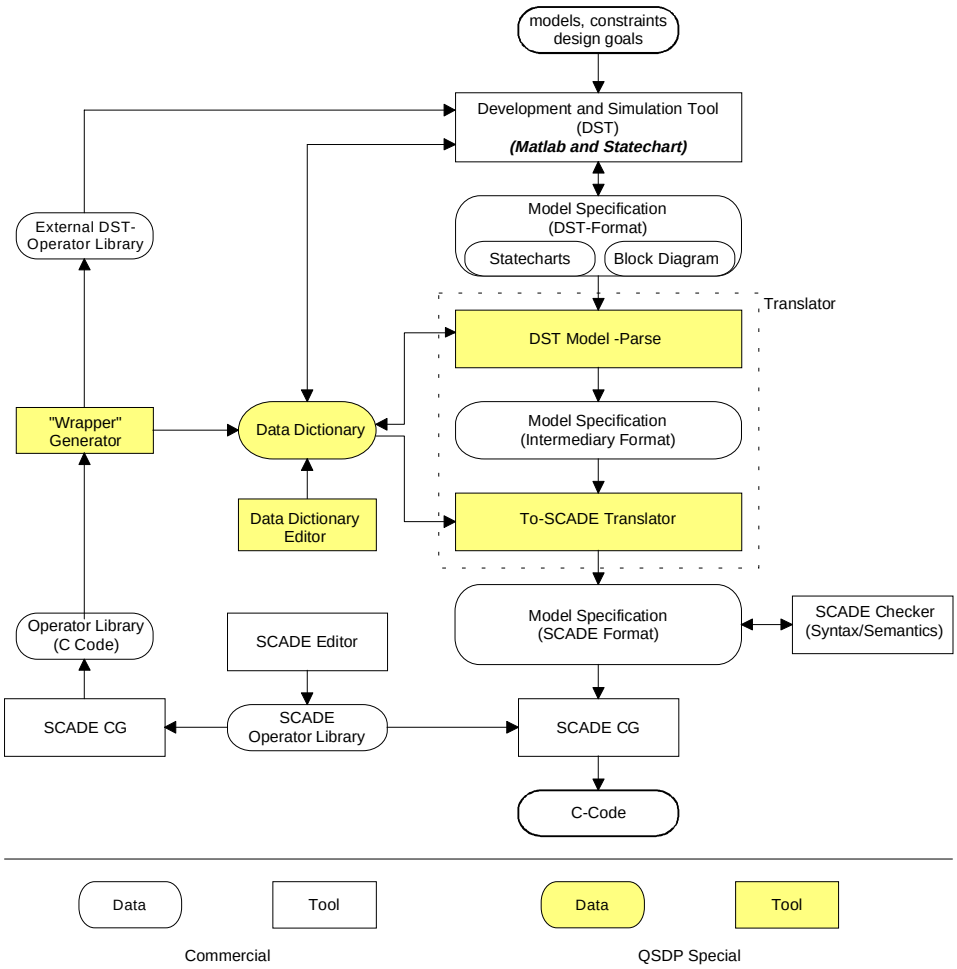

Bild 6: Qualified System Development Process

Mit dem qualifizierten **SCADE-Codegenerator** wird die Model Specification im SCADE Format unter Berücksichtigung der **SCADE Operator Library** in den **C-Code** umgewandelt, der mit gebräuchlichen Compilern in den Objekt-Code transformiert wird, der in die Hardware geladen werden kann.

4.3 Aktualisierung der DST-Operator Library

Parallel zur eigentlichen Entwicklung ist in Bild 6 von unten und nach oben die Aktualisierung der **DST-Operator Library** dargestellt.

SCADE nutzt eine Bibliothek mit einfachen Basisoperatoren. Für effizientes Arbeiten ist zusätzlich eine Bibliothek mit speziellen vom Anwender zu formulierenden Operatoren notwendig. All diese Operatoren stehen in der **DST Operator Library** als externe Operatoren für die Spezifikation der Gerätefunktionalität in Matlab zur Verfügung. Dazu werden die Operatoren der **SCADE Operator Library** mit dem **SCADE Code-Generator** nach C übersetzt und durch einen **Wrapper-Generator** mit einem Rahmen versehen, der ihren Einsatz als externe Operatoren in Matlab ermöglicht. Diese Übersetzung kann zur Aufwandsreduzierung z.B. mit einem Makefile und weiteren Werkzeugen mechanisiert werden. Sie wird zentral verwaltet, d.h. alle Nutzer greifen auf dieselbe Bibliothek zu. Durch Automatisierung wird ferner erreicht, daß die DST-Operator Library immer auf dem aktuellen Stand der SCADE-Operator Library gehalten wird. Diese Übertragung der Library ist als Arbeitsvorbereitung für neue Projekte zu sehen, die nach jeder Änderung der SCADE-Operator Library durchgeführt werden muß. Dies sollte aber innerhalb eines Projektes vermieden werden, da eine Bibliothekserweiterung eine neue Zertifizierung erfordert. Neue Operatoren werden mit dem **SCADE-Editor** erstellt.

4.3 Prototypische QSDP-Realisierung

Im Vorfeld der prototypischen Realisierung wurden die im Bild 6 hinterlegten QSDP-Software-Tools und das Data Dictionary detailliert spezifiziert. Bei der prototypischen Realisierung des QSDP wurde besonderer Wert auf die Nutzung kommerziell verfügbarer Tools gelegt. Im Einzelnen sind dies:

- DST-Model Parser
- Data Dictionary Editor

Kommerziell nicht erhältliche sind:

- To-SCADE Translator
- Wrapper Generator

Diese Tools wurden prototypisch realisiert und zusammen mit einer Matlab/Simulink-Entwicklungs- und Simulationsumgebung und dem SCADE-Codegenerator zu einem funktionsfähigen Qualifiable System Development Process konfiguriert.

Die Spezifikation der Gerätefunktionalität der Avionik-Komponenten in Matlab erfolgt dabei nur unter Nutzung der externen SCADE-Operatoren. Damit ist die Übertragbarkeit dieser Spezifikation nach SCADE gesichert. Für die Modellierung der Umgebung und der Simulations-Stimuli, die Auswertung usw. kann dagegen die volle Werkzeugfunktionalität von Matlab benutzt werden.

Von besonderer Bedeutung ist der **Wrapper-Generator.** Diese für QSDP charakteristische Funktion ist maßgeblich für die Durchgängigkeit und Qualifizierbarkeit des gesamten Prozesses. Der Wrapper-Generator macht den Entwicklungsprozeß ohne Einbussen in der Leistungsfähigkeit der Systemsimulation dadurch qualifizierbar, daß er die einfachen und damit qualifizierbaren SCADE-Operatoren und Datenstrukturen der qualifizierbaren SCADE-Operator Library auch für die Development- und Simulationstools (DST) verfügbar macht.

Die Systemsimulation kann so mit exakt den Steuer- und Regelalgorithmen arbeiten, die bei einer automatischen Codegenerierung auch in die Zielhardware geladen wird. Der Codegenerator bekommt andererseits von der Systemsimulation nur die qualifizierbaren Operatoren und Datenstrukturen übergeben.

Der **Wrapper-Generator** bindet zusätzliche Funktionalität an die Ein- und Ausgänge der Komponenten, um die Umwandlung der Datentypen an die Schnittstellen zu gewährleisten. Bei der Übertragung durch den Wrapper werden Informationen bezüglich der Datentypen der Ein- und Ausgänge in das Data Dictionary eingetragen. Der Data-Dictionary-Editor bewirkt die automatische Eintragung der Daten und ermöglicht zugleich die Dokumentation z.B. von Dimensionen, Bereichsgrenzen, usw..

Ein zweistufiger **Translator** übersetzt die Gerätespezifikation unter Nutzung der **Data-Dictionary** Informationen nach SCADE. Mit Hilfe des SCADE Checkers kann die Konsistenz der erzeugten SCADE-Spezifikation und damit der Gerätespezifikation in Matlab geprüft werden. Eine fehlerhafte Verwendung nicht zugelassener Matlab-Operatoren in der Gerätespezifikation oder Inkonsistenzen im Data-Dictionary werden spätestens an dieser Stelle offenbar. Sind keine Inkonsistenzen mehr enthalten, kann aus der SCADE-Spezifikation mit dem qualifizierten SCADE-Code-Generator C-Code erzeugt werden. Dieser kann nach Compilierung auf der Zielhardware verwendet oder für Simulationszwecke als komplettes Modul in Matlab eingebunden werden.

Um die Abbildung der Spezifikation in SCADE (und im später generierten Code) detailliert zu beeinflussen, werden automatische Abbildungs-vorschriften für die Variablen in das **Data Dictionary** gelegt. Sie beschreiben u.a. die Datentypen der Matlab-Variablen nach der Umsetzung in SCADE. Weitere Informationen im Data Dictionary werden für die Zertifizierungsunterlagen benötigt.

5 EVALUIERUNG

Zur Evaluierung der prototypisch realisierten QSDP-Entwicklungsumgebung wurde in Abstimmung mit dem Projektträger und Dasa Airbus das Wasserballastsystem als flugzeugrelevantes Anwendungsbeispiel aus dem Verantwortungsbereich von Dasa Airbus ausgewählt. Das Wasserballastsystem ist ein Teil des Flight Test Equipments. Es dient während der Flugerprobung zur Einstellung unterschiedlicher Schwerpunktlagen für die Analyse der Eigenschaften des Flugzeugs und der Flugsteuerung.

Zu diesem Zweck werden in den zu erprobenden Flugzeugen Wasserballasttanks verteilt. Die Wasserballaststeuerung steuert durch Umpumpen von Wasser zwischen den verschiedenen Tanks die Schwerpunktlage des Flugzeugs. Die Steuerung ist insofern sicherheitskritisch, als eine unkontrollierte Verschiebung des Schwerpunktes das Flugzeug in eine aerodynamisch instabile Lage bringen kann.

Die Evaluierung des QSDP-Pozesses lief in mehreren Schritten ab. Zunächst wurde mit Hilfe des prototypischen Wrapper Generators die SCADE-Library in Matlab/Simulink aktiviert. Aufbauend auf dieser Bibliothek haben Systemingenieure von Dasa Airbus die Prozeßmodelle des Wasserballastsystems und die dazu gehörigen Steuer-, Regel- und Analysefunktionen in Matlab formuliert und simuliert.

Die Steuerfunktionen wurden mit Statecharts beschrieben, während die Regelungsfunktionen in Form von Blockschaltbildern spezifiziert wurden.

Die Entwickler des QSDP-Prozesses haben daraufhin erfolgreich die Umsetzung der in Matlab simulierten Steuerfunktionen in den automatisch mit SCADE erzeugten C-Code gestartet, der mit einem bekannten C-Compiler in den Objektcode gewandelt wurde. Dieser wurde als ablauffähiger Code auf den Zielrechner geladen und seine Funktionsfähigkeit im Testrig des Wasserballastsystems getestet.

6 ZUSAMMENFASSUNG UND AUSBLICK

Mit QSDP wurde eine Entwicklungsumgebung definiert und prototypisch realisiert, mit deren Hilfe zertifizierbare Software für Steuer- und Regelsysteme

in Flugzeugen weitgehend automatisch aus grafischen Spezifikationen generiert werden kann. Diese Entwicklungsumgebung soll Grundbestandteil einer zukünftigen von der Spezifikation bis zum Objekt-Code lückenlosen Werkzeugkette sein. Die Funktionsfähigkeit des Prozesses ist demonstriert.

Die kommerzielle Realisierung eines solchen Prozesses erfordert eine enge Abstimmung zwischen Dasa Airbus und den europäischen Airbus-Partnern, den Airbus-Zulieferern, DaimlerChrysler Forschung und nicht zuletzt den kommerziellen Software-Herstellern. Diese müssen ihre verfügbaren Tools mit Blick auf den Qualified System Development Process überarbeiten, und ggfs. Brücken zu den Softwaretools anderer SW-Hersteller bauen. Die Airbus-Partner und ihre Zulieferer müssen gemeinsame Vorgehensweisen und Schnittstellen im Entwicklungs- und Fertigungsprozeß vereinbaren, um für die SW-Hersteller eine stabile wirtschaftliche Basis bieten zu können.

Deshalb wurde die Diskussion mit kommerziellen SW-Herstellern und verschiedenen Partnern aus der Luftfahrtindustrie aufgenommen. Zur Zeit ist zusammen mit den Airbus-Partnern das im 5. Rahmenprogramm geplante SafeAir Projekt in Vorbereitung, das die Realisierung eines qualifizierbaren automatischen Codegenerators im Umfeld eines " Avionic System Development Environment" zum Ziel hat.

7. Dank

Wir danken Gisela John, Matthias Hoffmann, Eustathios Aposporidis, Jan-Peter Munk, Dr. Jörg Janning und Dr. Notker Amann von der DaimlerChrysler Forschung für Ihre konstruktiven Projektbeiträge, Kai Ottilige von Dasa Airbus für die Unterstützung bei der Evaluierung und Dr.Hubertus Schmidtlein, Dr.Adrian Hintz und Rüdiger von Mandel vom EFCS-Team der Dasa Airbus für ihre Anregungen und Diskussionsbeiträge.

8 LITERATUR

8.1 Standards und Normen

[1.1] *RTCA/DO178-B, Software Considerations in Airborne Systems and Equipment Certification. Requirements and Technical Concepts for Aviation, Inc. (RTCA), Dezember 1992.*

[1.2] *Airbus Directives and Procedures ABD 100, Issue A: Equipment Design – Requirements for Suppliers. Airbus Industrie, Toulouse, Frankreich, 1995*

[1.3] *JAR 25 – Joint Airworthiness Requirements, Part 25: Large Aeroplanes, Change 13. Joint Airworthiness Authority, 1990*

[1.4] *ARINC Report 652: Guidance for Avionics Software Management. ARINC, Anapolis, MY, USA, 1993*

8.2 Aufsätze

[2.1] *R. Luckner, T. Heintsch, Developement of Manual Flight Control Functions for a Small Transport Aircraft, ICAS–94–7.6.1, 1994.*

8.3 Softwaredokumentation

[3.1] *KRÖGER, A.: Data Flow Oriented Control Law Design with the Graphical Language HOSTESS. Deutsche Aerospace Airbus Gmbh, Germany.*

[3.2] *INTEGRATED SYSTEMS INC., introducing MATRIX X Version 6.0, (Herstellerinformation 02/97), 1997*

[3.3] *SAO+ Environment: Specification and Design of Reactive Systems. Verilog SAO+ Trainingsunterlagen, Berlin, 1997*

[3.4] *SAO+/SAGACG C Code Generator Reference Manual, Version 2.1(Release 2.1.3, 1996). Verilog SA, Paris, Aug. 1992.*

[3.5] *SAO+/DF Language Reference Manual, Version 1.3.2. Verilog SA, Paris, Dec. 1996.*

[3.6] *SCADE Release Note, Version 2.1. Verilog SA, Paris, Jan. 1998.*

[3.7] *THE MATHWORKS INC., Matlab based Books, 1996*

[3.8] *Using Simulink (Version 2): Dynamic System Simulation for MATLAB. The Mathworks, Natick, MA, Jan. 1997.*

[3.10] *Getting Started with MATLAB (Version 5): The Language of Technical Computing. The Mathworks, Natick, MA, May 1997.*

[3.11] *AdaCover an Coverage Tool for Certification, Thomson Software Products, 1994*

[3.12] *UML Unified Modeling Language, Version 1.0, Januar 1997, http://rational.com*