Entwicklung von "High Quality Embedded Systems" in der industriellen Praxis

CASE Anwendertag 1995

DLR, Göttingen

19. September 1995

Rainer Gerlich

email: gerlich@t-online.de

ENTWICKLUNG VON "HIGH QUALITY EMBEDDED SYSTEMS" IN DER INDUSTRIELLEN PRAXIS

Rainer Gerlich
email: gerlich@t-online.de

1. EINLEITUNG

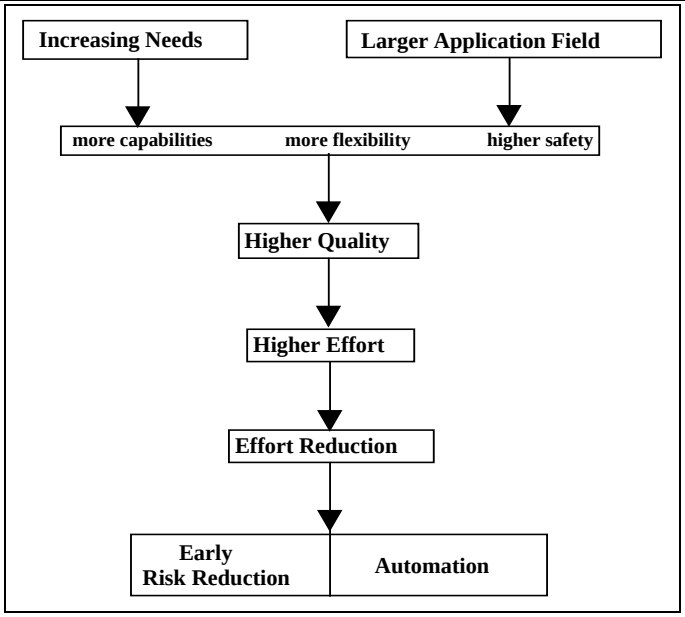

Bild 1: Steigende Anforderungen vs. Effizienz

Echtzeitsysteme werden heute auf vielen Gebieten zur Steuerung und Überwachung eingesetzt, auch zunehmend in sicherheitskritischen Bereichen und in Gebieten, für die hohe Zuverlässigkeit verlangt wird. Steigende Anforderungen (Bild 1), z.B. der Wunsch nach mehr Funktionalität und mehr Bedienkomfort, führen zu immer größerer Komplexität. Gleichzeitig steigt zur Abdeckung der wachsenden Bedürfnisse der Softwareanteil in Echtzeitsystemen kontinuierlich an.

Aus der Vergangenheit sind hinreichend die Probleme bekannt, die bei der Entwicklung von Software, und Echtzeitsoftware im besonderen, zu bewältigen sind. Je mehr Software eingesetzt wird, desto größer sind ihre Auswirkungen, insbesondere bei fehlerhafter Software. Um zunehmende Probleme bei wachsenden Softwareanteilen zu vermeiden, ist in Zukunft mehr Qualität notwendig. Von besonderem Interesse ist dies natürlich für sicherheitskritische Anwendungen.

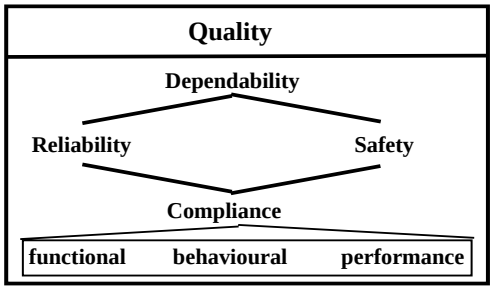

Bild 2: Qualitätsmerkmale

Qualität impliziert Zuverlässigkeit und Sicherheit neben der gewünschten Funktionalität, Verhalten und Performance (Bild 2). Höhere Qualität bedeutet mehr Aufwand. Für den industriellen Einsatz ist der Aufwand (im Sinne von Kosten und Zeit) neben den technischen Aspekten ebenso eine kritische Größe bei der Systementwicklung. Denn wird der Aufwand zu hoch, dann ist die Entwicklung unter Umständen wegen des ungünstigen Kosten/Nutzen-Verhältnisses wirtschaftlich uninteressant.

Das Entwicklungsrisiko ist eine weitere kritische Größe. Hiermit ist das Risiko gemeint, das Entwicklungsziel eventuell nicht oder nur teilweise zu erreichen

Aus der sichtbaren "Form" der Software, dem Sourcecode, kann man nicht auf ihre Korrektheit schließen. Ob Software sich wie gewünscht verhält oder nicht, kann man erst feststellen, wenn die von der Software produzierten Ergebnisse dem Menschen anschaulich vermittelt werden. Dies geschieht zur Zeit aber erst zu einem recht späten Zeitpunkt im

Systementwicklungszyklus (Bild 3), beispielsweise wenn nach dem sogenannten V-Modell vorgegangen wird erst bei Test und Integration.

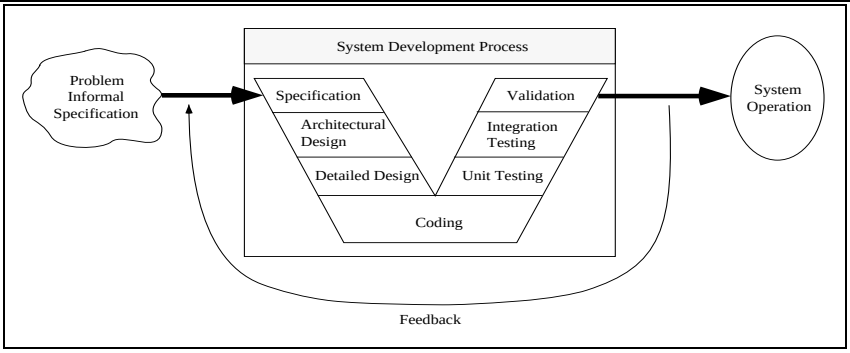

Bild 3: Entwicklungsrisiko beim V-Modell

Hardwareausfälle entstehen hauptsächlich durch (statistisch auftretende) Degradierung des Materials, weniger durch (systematisch auftretende) Spezifikations- und Entwurfsfehler. Software kann nicht degradieren. Ausfälle durch Software beruhen allein auf Spezifikations-, Entwurfs- oder Codierungsfehlern.

Hardwareausfällen durch Degradierung kann man durch Redundanz begegnen. Bei Spezifikations-, Entwurfs- und Codierungsfehlern hilft jedoch nur sorgfältiges Vorgehen sowie die Verwendung von geeigneten Hilfsmitteln, die die Aufdeckung von Fehlern erlauben und erleichtern. Dazu gehören die formale Verifikation eines Systems und seine Validierung durch Ausführung und Visualisierung der Ergebnisse.

Um für die steigenden Aufgaben gerüstet zu sein, ist es daher notwendig, die Schwachstellen der bisherigen Entwicklungsmethoden hinsichtlich Risiken und Aufwand aufzudecken und Alternativen zu entwickeln. Aufgabe dabei ist auch, nicht nur die einzelnen Entwicklungsphasen zu verbessern, sondern ein ganzheitliches Konzept zu erarbeiten, das den gesamten Entwicklungszyklus optimiert. Dazu gehört auch die Aufgabenverteilung zwischen Mensch und Entwicklungswerkzeugen.

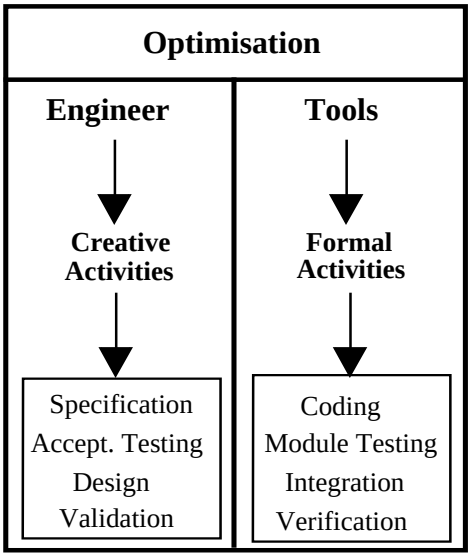

Bild 4: Optimierung der Aufgabenverteilung

Auch in der Raumfahrt steigt die Komplexität der Systeme. Hohe Zuverlässigkeit und Sicherheit werden in diesem Anwendungsbereich schon immer gefordert. Aber auch Kosten und das Entwicklungsrisiko müssen beachtet und auch zukünftig im akzeptablen Rahmen gehalten werden. Daher hat die ESA Aktivitäten initiiert, durch die neue System- und Softwareentwicklungsmethoden definiert wurden, um die anstehenden Probleme zu bewältigen.

Im Rahmen der Pilotprojekte HRDMS [1] und OMBSIM [2] wurden neuartige Entwicklungsmethoden erprobt. Im Projekt OMBSIM wurde die Methode "EaSyVaDe" definiert, die einen systematischen Ansatz hinsichtlich frühzeitiger Validierung, inkrementeller Vorgehensweise und der Entwicklung von Hardware-Software (Echtzeit-) Systemen (Embedded Systems) einführt. In einem weiteren Projekt DDV [3] wird EaSyVaDe in einem neuen Umfeld eingesetzt. EaSyVaDe (Early System Validation of Design) [4] ist das

Ergebnis einer Reihe von Vorentwicklungen, Untersuchungen und Erfahrungen in Projekten, die seit 1986 durchgeführt wurden.

EaSyVaDe ist aber nicht nur für Raumfahrtanwendungen interessant und einsetzbar. Auch kleinere Projekte oder Anwendungen aus anderen Bereichen können von dieser Entwicklungsstrategie profitieren. Die Methode EaSyVaDe wird durch die Entwicklungsumgebung EaSySim unterstützt.

2. FRÜHZEITIGE UND SCHRITTWEISE VALIDIERUNG ALS LÖSUNG

Um höhere Effizienz zu erreichen, ist eine Optimierung beim Einsatz von Ingenieuren und Tools wünschenswert (Bild 4). Hierzu ist eine stärkere Konzentration des Ingenieurs auf die kreativen Bereiche der Systementwicklung erforderlich, während der Einsatz von Tools dort intensiviert werden muß, wo eine Formalisierung und Automatisierung möglich ist. Dies führt zum stärkeren Einsatz eines Ingenieurs in frühen Entwicklungsphasen: das Entwicklungsrisiko wird verringert und der Automatisierungsgrad erhöht (Bild 5). Zur Zeit ist es eher umgekehrt: der Hauptaufwand für einen Ingenieur fällt bei der Codierung, beim Testen und Integrieren an, und zwar in späten Entwicklungsphasen.

Dies führt aus folgenden Gründen zur Qualitätssteigerung:

- Durch schrittweise und frühzeitige Validierung sind ständig die aktuellen Systemfähigkeiten bekannt, sowohl funktional als auch performancemäßig. Mit den ausführbaren Modellen können Simulationen unter nominalen und Fehlersituationen durchgeführt werden. Das Risiko, das Entwicklungsziel zu verfehlen, verringert sich hierdurch.
- Durch Einsatz formaler Methoden können Diskrepanzen automatisch mit Tools entdeckt werden, wie fehlerhafte Schnittstellen oder fehlerhaftes Verhalten, und die Codegenerierung kann automatisiert werden. Dadurch steigt die Effizienz der Systementwicklung.

Je höher die Formalisierung ist, desto größer ist auch der Bereich, der durch Tools effizient abgedeckt werden kann.

Risk Reduction	**Automation**
executable models (simulation)	formal methods
	automated checks
refinement stepwise hierarchical	code generation from simulation
simulation nominal conditions non-nominal conditions	tool integration
	tool improvement (add-on's)
homogeneous system approach hardware-software co-design	
late hardware-software trade-off	
Increased Reuse	

Bild 5: Strategie zur Qualitäts- und Effizienzsteigerung

Insgesamt erhält man bessere Qualität, weil mehr Entwicklungsfehler frühzeitig und automatisch entdeckt werden können. Durch automatische Codegenerierung aus den bereits validierten Modellen können keine neuen Fehler beim Codieren erzeugt werden.

Die Wiederverwendung von Systemkomponenten steigt, weil durch die formalisierte Vorgehensweise Komponenten mit ähnlicher Funktionalität frühzeitig identiffiziert und harmonisiert werden können, bzw. schon vorhandene Komponenten besser und frühzeitig integriert werden können.

EaSyVaDe hilft, Entwicklungsrisiken frühzeitig zu identifizieren, indem ausführbare Spezifikations- und Entwurfsmodelle

verwendet werden. Denn nur beim Ausführen der Software kann man feststellen, was spezifiziert oder entworfen wird und ob dies auch den Erwartungen entspricht. Hierbei wird von EaSyVaDe ein kontinuierlicher und konsistenter Übergang zwischen Spezifikation und Entwurf ermöglicht.

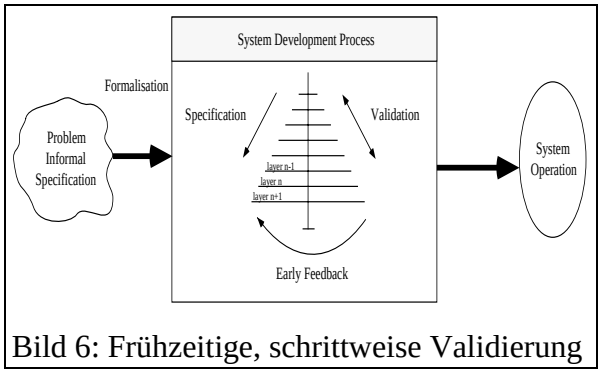

Bild 6: Frühzeitige, schrittweise Validierung

Nach jedem Verfeinerungsschritt ist eine funktionelle und performancemäßige Überprüfung durch Ausführung der Modelle möglich: eine vorläufige Validierung von Spezifikation und Design kann somit in einem frühen Entwicklungsstadium durchgeführt werden. Hierbei sind Iterationen möglich und sehr wahrscheinlich. Wenn das Systemverhalten genügend gut verstanden wurde und schließlich den Erwartungen entspricht, wird auf der nächsten Hierarchieebene weiter verfeinert und validiert (Bild 6).

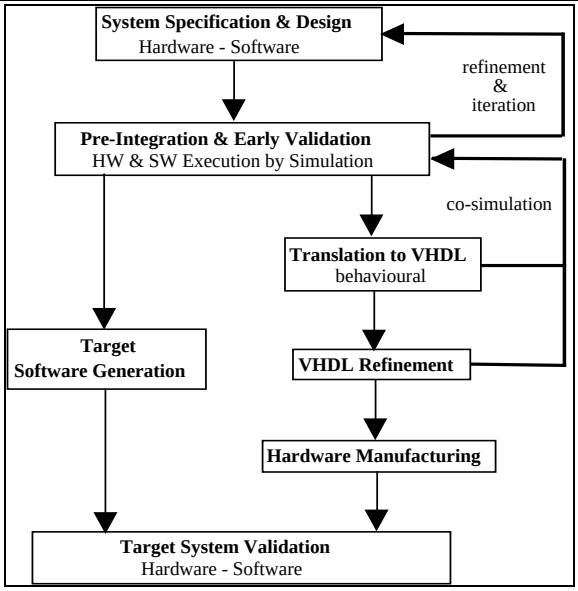

Bild 7: Homogener Ansatz für Systementwicklung

Die Verfeinerung erfolgt schrittweise und hierarchisch. Auf diese Weise wird das System inkrementell entwickelt und Schritt für Schritt validiert. Die frühzeitige Validierung beruht auf der Annahme, daß die spezifizierten Anforderungen an die unteren Hierarchieebenen erfüllt werden können. Insofern ist immer noch ein Entwicklungsrisiko vorhanden, bis die unterste Ebene erreicht wird. Da jedoch die Anforderungen bekannt sind, ist die Wahrscheinlichkeit größer, Fälle, die nicht realisierbar sind, zu erkennen als wenn die Anforderungen nur sehr vage oder überhaupt nicht bekannt sind. Dies trifft insbesondere für Performance und Ressourceanforderungen zu.

Die Modelle repräsentieren Systemkomponenten, nicht Hardware oder Software (Bild 7). Dadurch kann die Entscheidung "Hardware oder Software?" solange verzögert werden, bis die Anforderungen bekannt sind. Zur Zeit muß diese Entscheidung frühzeitig getroffen werden. Eine Revision der Entscheidung ist später nur noch selten möglich, und wenn, dann nur mit relativ hohem Aufwand.

Alle Systemkomponenten können in einer homogenen Umgebung ausgeführt und validiert werden. Aus Softwarekomponenten kann dann direkt Targetcode erzeugt werden. Für Komponenten, die in Hardware realisiert werden sollen, kann die Spezifikation "Behavioural-VHDL" übersetzt werden. Zur Zeit werden Arbeiten [5,6] zur Umsetzung von SDL [7] nach VHDL [8] durchgeführt. Nach der Umsetzung kann in VHDL weiter verfeinert werden. Da Toolhersteller in Zukunft bessere Schnittstellen zur Co-Simulation anbieten werden, könnte auch eine Co-Simulation mit verfeinerten VHDL-Modellen möglich werden.

Die Validierungsaktivitäten bestehen aus zwei Teilen (Bild 8)

- der Verifikation von statischen und dynamischen Eigenschaften durch Tools, und
- der Validierung des operationellen Verhaltens unter nominalen und fehlerhaften Bedingungen durch den Ingenieur unterstützt durch Tools.

Durch Formalisierung können die Verifikationsschrittte automatisch von Tools durchgeführt werden, während die Validierung Aufgabe des Ingenieurs ist unterstützt durch Tools für die Ausführung und Ergebnisauswertung.

Principal System Validation Steps	
verification (tools)	checks by formalisation static properties (interfaces, types, ...) dynamic properties (behaviour, state transitions, ...)
validation (engineer)	system execution nominal conditions non-nominal conditions

Bild 8: Verifikation und Validierung

Insgesamt werden somit die Einwirkungs- und Kontrollmöglichkeiten in den frühen Entwicklungsphasen verstärkt. Tools werden dort eingesetzt, wo ihr Nutzen am größten ist. Dies führt insgesamt zu besserer Qualität bei höherer Effizienz.

3. ZUSAMMENFASSUNG

EaSyVaDe erlaubt durch frühzeitige Validierung, Formalisierung und Automatisierung die effiziente Entwicklung von Echtzeitsoftware hoher Qualität. Dies ist für sicherheitskritische Anwendungen und Anwendungen wichtig, für die hohe Zuverlässigkeit und Verfügbarkeit gefordert wird, aber auch für andere Anwendungen, wenn Entwicklungszeit und -kosten gesenkt werden sollen. Der Einsatz eines Entwicklungsingenieurs wird stärker auf die kreativen und kritischen Entwicklungsaktivitäten verlagert. Für automatisch durchführbare Aktivitäten wie Konsistenz-Prüfungen und Codegenerierung werden dagegen Tools eingesetzt. Dazu werden verstärkt formale Methoden eingesetzt, auch für die Beschreibung des Echtzeit-Verhaltens.

EaSyVaDe erweitert die bisher auf regelungstechnische Anwendungen beschränkte Vorgehensweise [9,10], für Spezifikation und Entwurf formal definierte Standardkomponenten zu verwenden, auf die allgemeine System- und Softwareentwicklung und führt eine formale Vorgehensweise für die frühzeitige Validierung von Echtzeitsoftware ein.

EaSyVaDe unterstützt auch das Projektmanagement. Insbesondere berücksichtigt es die verteilte Entwicklung von Systemen, es nutzt ausführbare Modelle als Spezifikation und erlaubt die Re-Integration der durch einen Unterauftragnehmer verfeinerten Modelle.

Zu EaSyvaDe ist eine Entwicklungsumgebung EaSySim verfügbar, die die Methode unterstützt. EaSySim besteht aus den kommerziellen Tools GEODE [11] und SES/workbench [12] und Zusatzsoftware, die die beiden Tools miteinander verbindet und deren Funktionalität insgesamt erhöht, sodaß die Methode bestmöglichst unterstützt wird.

Es ist geplant, EaSyVaDe bei weiteren Projekten einzusetzen und EaSySim in eine existierende, umfassendere Entwicklungsumgebung zu integrieren.

In diesem Beitrag konnte nur ein kurzer Einblick in die gesamte Vorgehensweise gegeben werden. Wichtige Punkte wie die Performance oder Zuverlässigkeit von automatisch generiertem Code, der Übergang zwischen Spezifikation und Entwurf, die Berücksichtigung von Funktionalität und Performance und die Stabilität des Systems bei fortlaufender Verfeinerung konnten nicht oder nur kurz angesprochen werden. Es wird daher auf weitere Artikel [13,14,15,16] verwiesen, die auf Anfrage erhältlich sind.

REFERENZEN

[1] HRDMS (Highly Reliable DMS and Simulation), ESTEC contract no. 9882/92/NL/JG(SC), Final Report, 1994

[2] OMBSIM (On-Board Mangement System Behavioural Simulation, ESTEC contract no. 10430/93/NL/FM(SC), Report, OMBSIM/FR/001/DOR/, September 1995

[3] DDV (DMS Design Validation), ESTEC contract no. 9558/91/NL/JG(SC), Formal Methods and Tools, Selection & Justification Report, TR/171/PhH/95, 30.06.95

[4] R.Gerlich, V.Debus, Ch.Schaffer, Y.Tanurhan: EaSyVaDe: Early Validation of System Design by Behavioural Simulation, ESTEC 3rd Workshop on "Simulators for European Space Programmes" Noordwijk, November 15-17, 1994

[5] M.Lehmann: "Automatischer Übergang von SDL nach VHDL", Diplomarbeit, Fachhochschule Albstadt-Sigmaringen, September 1995

[6] T.B.Ismail, A.Jerray: "Synthesis Steps and Design Models for Codesign", IEEE Computer, February 1995, pp. 44-52

[7] ITU, Recommendation Z.100, Specification and Description Language, SDL, 1993. Blue Book, Vol. X.1, and appendices A, B, C, D, F1, F2, F3

[8] The Institute of Electrical and Electronics Engineers: "IEEE Standard VHDL Language Reference Manual (IEEE-1076-1992/B)", New York, 1993

[9] Matrix-X, Integrated Systems Inc., 3260 Jay Street, Santa Clara, CA 95054-3309, USA

[10] K.J.Vogel, D.H.Johnson: An Integrated COTS Prototyping and Test Environment for Spacecraft Control Systems, EUROSPACE Symposium on "Techology and Applications for Space Data Management Systems", January 25-27, 1994, Rome, Italy

[11] SDT, TeleLogic AB, PO Box 4128, S-20312 Malmö,Sweden

[12] SES/workbench, Scientific and Engineering Software Inc., Building A, 4301 Westbank Drive, Austin, Texas, 78746-6564, USA

[13] R.Gerlich, N.Schäfer, A.Schäferhoff: Early Validation of DSM Design by a Reusable Environment, EUROSPACE On-Board Data Management Symposium on "Technology and Applications for Space Data Management Systems", January 25-27, 1994, Rome, Italy

[14] R.Gerlich, V.Debus, Ch.Schaffer, Y.Tanurhan: EaSyVaDe: Early Validation of System Design by Behavioural Simulation, ESTEC 3rd Workshop on "Simulators for European Space Programmes" Noordwijk, November 15-17, 1994

[15] R.Gerlich, Th.Stingl, Ch.Schaffer, F.Teston, G.Martinelli: Use of an Extended SDL Environment for Specification and Design of On-Board Operations, Systems Engineering Workshop, ESTEC, Noordwijk, The Netherlands, November 28-30, 1995 (paper available by end of September 1995)

[16] R.Gerlich, C.Joergensen: An Alternative Lifecycle Based on Problem-Oriented Methods and Strategies, International Symposium on On-Board Real-Time Software, ESTEC, Noordwijk, The Netherlands, November 13-15, 1995 (paper available by end of September 1995)