6. Workshop Automatisierungstechnische Verfahren für die Medizin vom 24.-25. März 2006 in Rostock-Warnemünde

„Integrierte Entwicklung einer Steuerung für ein Beatmungsgerät“

Florian Dietz
Weinmann Geräte für Medizin GmbH+Co. KG, Hamburg, Deutschland
E-Mail: F.Dietz@weinmann.de

Band: Abstracts der Vorträge des 6. Workshops der Automed 2006
Editors: T. Ellerbrock
ISBN: 3-86009-296-0
Pages: 20-21

INTEGRIERTE ENTWICKLUNG EINER STEUERUNG FÜR EIN BEATMUNGSGERÄT

Florian Dietz
Weinmann Geräte für Medizin GmbH+Co. KG
22525 Hamburg

F.Dietz@weinmann.de

EINLEITUNG

Der Beitrag stellt Verfahren und Methoden zur Entwicklung einer mikrocontroller-basierten Firmware für ein Beatmungsgerät im Sinne des V-Modells dar. Die Integration der verwendeten Tools spielt eine große Rolle. Da die integrierte Entwicklung auch die Prozesse Simulation und Code-Generierung unter Matlab®/Simulink® umfasst, besteht ein wichtiger Teil des Vorgehens darin, die Sicherheit des generierten Codes zu gewährleisten. Der Beitrag umfasst neben einer Darstellung der verschiedenen Tools auch eigene Ansätze und Tools zur Integration. Ein Abriss über die erzielten Ergebnisse und Erfahrungen bildet den Abschluss des Beitrages.

SOFTWARE-ERSTELLUNG UND LEBENSZYKLUS

Die Entwicklung und der Lebenszyklus von Software für programmierbare elektrische medizinische Systeme (PEMS) sind vor allem in der DIN EN ISO 60601-1-4 normativ beschrieben [EN60601-1-4:2001]. Ziel dieser Norm ist die Gewährleistung der Software-Sicherheit für das betreffende Medizinprodukt über den gesamten Lebenszyklus. Qualitative Merkmale einer solchen Software sind beispielsweise Klarheit, Wartbarkeit, Reproduzierbarkeit und Rückverfolgbarkeit.

Um diese Ziele zu erreichen, sind bestimmte Prozesse bei der Software-Erstellung und -Pflege einzuhalten. Eine besondere Rolle spielt dabei das sogenannte V-Modell, wie es nicht nur in [ISO60601-1-4:2001] beschrieben wird, sondern auch in vielen anderen Bereichen der Entwicklung praktische Umsetzung erfährt. Einen Einblick in das V-Modell bieten der V-Modell-Browser der Fraunhofer Gesellschaft [V-Modell-Browser2006] sowie das V-Modell® XT der KBSt [V-Modell-XT2006].

Bezüglich der Software-Erstellung beschreibt das V-Modell zunächst den Weg der Dekomposition vom Anforderungsprofil bis zur Modulimplementierung und dann den Weg der Aggregation/Integration vom Modultest bis zur Anwendungsvalidierung.

SICHERUNGSMASSNAHMEN

Zur Qualitätssicherung einer Steuersoftware kommt mit zunehmender Komplexität der Systeme das statische Testen (Code-Review/-Walkthrough) immer weniger zum Einsatz, da die Gesamtkomplexität kaum erfasst werden kann. Das dynamische Testen der Software, d.h. während der Laufzeit, erlaubt Untersuchungen zur Funktion, Fehlerfreiheit und Robustheit. Nichtfunktionale Tests wie z.B. Laufzeitverhalten, Prozessorauslastung (Profiling), und Code-Abdeckung (Coverage) lassen sich nur am ausgeführten Zielsystem zuverlässig beurteilen. Insbesondere die Automatisierbarkeit und Reproduzierbarkeit von Testläufen ermöglichen erneute vollständige Tests nach jeder neuen Software-Revision.

Im Sinne des V-Modells werden parallel zum linken Schenkel bereits diejenigen Testkriterien und -verfahren festgelegt, die im rechten Schenkel des V-Modells während der Verifikation angewendet werden.

MODELLBASIERTE VERIFIKATIONSSCHRITTE

Mögliche Verfahren zur modellbasierten Verifikation werden im Folgenden erläutert:

Model-open-Loop (MoL): Erste Implementierung der Steuerung im Modell als ausführbare Spezifikation. Die Simulation des Modells wird durch Messvektoren stimuliert. Es können funktionale und nichtfunktionale Fehler wie z.B. Überläufe von Festkommagrößen aufgedeckt werden.

Model-in-the-Loop (MiL): Modell im Zusammenspiel mit einer Umweltsimulation als Basis für die weiteren Verifikationsschritte (Vergleich von Ergebnissen mit Back-to-Back-Tests, Fehlerinduktion etc.). Für die Umwelt kommen je nach Anforderung physikalische Modelle unterschiedlicher Komplexität in Frage. Erste Rückkopplungen zu den Anforderungen sind möglich. Unterschiedliche Implementierungen der Steuerungs-Modelle: Fließ-/Festkomma-Arithmetik, evtl. verschiedene Datentypen zur Berücksichtigung unterschiedlicher Prozessortypen.

Software-in-the-Loop (SiL): Kompilierte Software (z.B. nach automatischer Codegenerierung), die mit dem Umwelt-Modell auf dem Host-PC gekoppelt wird, um Abweichungen etwa durch Fehler in der Codegenerierung oder durch abweichende Datentyp-Definitionen zu entdecken. Durch die Verwendung anderer Compiler im Vergleich zum Mikrocontroller für SiL ist die Aussagefähigkeit dieser Tests eingeschränkt.

Rapid Prototyping (RP): Ausführung des Steuerungsmodells auf einer fließkommafähigen echtzeitfähigen Plattform. Über vielfältige Anschlüsse erfolgt die Anbindung der Umwelt (z.B. Elektronik, Mechanik). Somit können Funktionsmuster schnell in Betrieb genommen werden, ohne langwierige Anpassungen an bestimmte Prozessorarchitekturen, Festkommaarithmetiken oder Schnittstellen durchführen zu müssen. RP ist sinnvoll zur Datenakquisition, zur Parallelisierung von Testaufgaben und für Demonstrationen.

Processor-open-Loop (PoL): Kompilierung des Codes auf die Ziel-Hardware und Kopplung mit dem Host-PC. Stimulation des Mikrocontrollers mit Testvektoren. Hier kann die gesamte Toolkette zur Binär-Code-Erzeugung auf Fehler überprüft werden (Back-to-Back).

Processor-in-the-Loop (PiL): Wie PoL, jedoch Kopplung mit dem Modell der Umwelt auf dem Host-PC. Diese Form der Kopplung ist schwierig umzusetzen, da die Echtzeitfähigkeit oftmals durch die Rechenleistung des Host-PC und/oder die langsame Datenverbindung eingeschränkt ist.

Hardware-in-the-Loop (HiL): Ähnlich zu PiL wird die Zielhardware mit einem Umwelt-Modell gekoppelt. Dieses wird jedoch auf einem Echtzeitsystem mit umfangreichen Schnittstellen implementiert (vgl. RP). Vorteilhaft im Vergleich zu PiL ist insbesondere die Berücksichtigung aller Hardware-Schnittstellen und Treiber des Zielsystems. Back-to-Back-Tests müssen aufgrund der Hardware-Kopplung mit einer gewissen Toleranz bewertet werden.

Mit der Produkt-Implementierung kann schließlich die System-Validierung erfolgen.

Tab. 1 bietet eine Übersicht über die Testsysteme und die jeweils eingesetzten Implementierungsebenen für Steuerung und Umgebung sowie deren Kopplung.

Tab. 1: Übersicht der Testebenen: Implementierung von Steuerungssoftware und Umgebungsmodell

Test-System	Steuer-Software	Steuer-Prozessor	Verbindung zur Umgebung	Umgebung
MoL	Modell	Host-PC	Daten-V.	Stimuli
MiL	Modell	Host-PC	Daten-V.	Modell
SiL	Code PC	Host-PC	Daten-V.	Modell
RP	Code RP-Syst.	RP-System	Dig.+anal. Schnittst.	Ziel-Mechatronik
PoL	Code µC	µC	Daten-Schnittst.	Stimuli
PiL	Code µC	µC	Daten-Schnittst.	Modell
HiL	Code µC	µC	Dig.+anal. Schnittst.	Echtzeitsimulation
Produkt	Code µC	µC	Dig.+anal. Schnittst.	Ziel-Mechatronik

EINGESETZTE TOOLKETTE

Nach der Modellierung der Steuerung und der Umwelt unter Simulink erfolgt die Codegenerierung mittels des Embedded Coders der Fa. MathWorks (inkl. Laufzeitumgebung) und anschließend die Kompilierung auf das Zielsystem.

Bis auf RP und PiL werden in der laufenden Entwicklung alle Ausprägungen der Verifikation eingesetzt, um die Funktionalität der Software abzuprüfen. Zusätzlich werden auf der Ebene MiL noch weitere Aspekte wie Coverage, Profiling, Einhaltung von Modellierungsrichtlinien und Regelgütekriterien getestet. Dazu wird ein Tool der Fa. Weinmann zum automatisierten Test der genannten Kriterien eingesetzt (Abb. 1).

Mit einem programmierbaren Lungensimulator werden schließlich mit dem Produkt Back-to-Back-Tests durchgeführt.

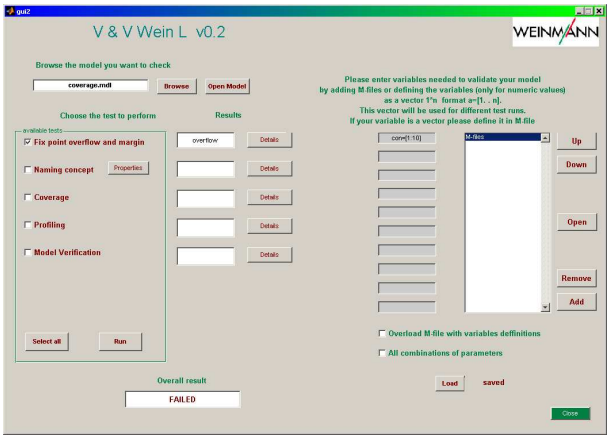

Abb. 1: Tool zum Test nichtfunktionaler Kriterien

ERGEBNISSE UND ERFAHRUNGEN

Bereits mit der Einführung der Werkzeuge und Methoden ist eine Zeitersparnis in der Entwicklung gegenüber nicht-modellbasierten Ansätzen erkennbar. Durchgängige Back-to-Back-Tests können bis zum fertigen Produkt ausgeführt werden. Fehler in der Codegenerierung konnten trotz der Komplexität der Steuerung bislang nicht festgestellt werden.

LITERATURHINWEISE

[Lamberg2005]
K. Lamberg, M. Beine, „Testmethoden und -tools in der modellbasierten Funktionsentwicklung“ in *Proceedings der Jahrestagung der ASIM/GI Fachgruppe 4.5.5 ‚Simulation technischer Systeme’*, Berlin, 2005, ISSN 1436-9915

[EN60601-1-4:2001]
DIN EN 60601-1-4:2001, *Medizinische elektrische Geräte – Programmierbare elektrische medizinische Systeme*, Beuth, Berlin, 2001

[V-Modell-Browser2006]
http://www.iese.fhg.de/VModell. Fraunhofer IESE, Kaiserslautern, 2006

[V-Modell-XT2006]
http://www.v-modell-xt.de. Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung, Berlin, 2006