Ansätze zum Nachweis der Gleichwertigkeit von Software-Komponenten

Hans-Werner Wiesbrock
IT Power Consultants, 13355 Berlin, Germany
hans-werner.wiesbrock@itpower.de

Abstract: Wie kann man heute ein bereits seit Jahren im Markt befindlichen System nach einer Änderung effizient rezertifizieren?

In diesem Artikel wird ein Lösungsansatz skizziert. Ausgehend von einer Risikoanalyse werden geeignete Vorgehensweisen und Kriterien abgeleitet, mit denen sich operational die Gleichwertigkeit von Software-Komponenten fassen lässt. Ein effizientes Vorgehen sähe dann wie folgt aus: Abgestimmt mit der Zertifizierungsbehörde werden für die zu ersetzenden Komponenten anhand eines Risikomodells die geeigneten Kriterien ihrer Gleichwertigkeit definiert. Der Hersteller weist anschließend nach, dass die geforderte Äquivalenz gilt.

1 Herausforderung

Mit dem Vordringen Software-intensiver, eingebetteter Systeme in sicherheitsrelevante Bereiche steigen die Anforderungen an Sicherheit und Zuverlässigkeit. Damit wachsen auch die Aufwände, um die qualitätsabsichernden Zulassungsverfahren zu passieren. Lange Produktzyklenzeiten und die Entwicklung neuer Technologien andererseits drängen auf partielle Ersetzungen oder Aufwertungen bestehender Systeme. Der Austausch einer Softwarekomponente kann erforderlich sein, wenn sich neue Programmierparadigmen und -architekturen bewähren und durchsetzen, die modellbasierte Vorgehensweise gegenüber dem herkömmlichen Codieren Einzug findet, verbesserte Kommunkationsstandards alte Standards ersetzen oder alte Compiler nicht mehr gewartet werden. Wie in vielen Bereichen können notwendige und erprobte Richtlinien und mit ihnen häufig verbundene Zertifizierungsprozesse hier auch zum Hemmschuh für sinnvolle Neuentwicklungen und nutzbringende Erweiterungen werden.

> *Medizinprodukte sind auf Grund der hohen Anforderungen an Sicherheit und Zuverlässigkeit durch besonders lange Produktlebenszyklen und extrem aufwändige Zulassungsverfahren gekennzeichnet. Eine besondere Rolle hierbei nehmen softwarebasierte Systeme ein. Auf Grund der seit Jahren stetig zunehmenden normativen Anforderungen stellen sich zwei grundlegende Fragen:*
>
> - *Wie kann man heute ein bereits seit Jahren im Markt befindliches System nach einer Änderung effizient rezertifizieren?*

- *Wie muss ein System heute entwickelt werden, um in der Zukunft bei Änderungen effizient rezertifiziert werden zu können?*

[SPE09]

Diese Fragestellungen haben enge Parallelen in der allgemeinen Systementwicklung. Wie kann man ein System erstellen, welches flexibel auf Änderungen reagieren kann? Die objektorientierte Analyse (OOA) antwortete hier mit Kapselung. Es sind geeignete Komponenten zu identifizieren, in der möglichst weitgehend die eigentliche Funktionalität verborgen ist [Bal96]. Entwurfsmuster unterstützen dabei den Entwickler [GHJV95, BMR+98]. Dann lassen sich einzelne, gekapselte Komponenten kontrollierbar austauschen. Allerdings sind im medizintechnischen Bereich neben einer objektorientierten Analyse zusätzlich die Sicherheitsmodelle zu berücksichtigen. Meist werden hierzu Fehlerbaumanalyse oder FMEA eingesetzt. In der deduktiven Fehlerbaumanalyse (FTA) geht man von einem Fehlverhalten aus und untersucht,wie es dazu kommen kann. Dazu komplementär ist die induktive Fehlerursache und -wirkungsanalyse, (FMEA), bei der aus möglichen Ausfällen im System deren Wirkung untersucht wird. Diese Sicherheitsanalysen sind auf die einzelnen Komponenten herunterzubrechen und wirken ihrerseits auf die gewählten Architekturen zurück [KLM03, GTS04].

Wie lässt sich sicherstellen, dass beim Ersetzen nicht unerwartete Nebeneffekte auftreten? Auch hier gibt es Standardmethoden: Back-To-Back-Tests oder Regressionstests, d.h., die neue Version wird gegen die alte Version evaluiert. Im Kontext sicherheitsrelevanter Alt- und Neusysteme ist hierbei jedoch offen, wie diese Tests aufzusetzen, welche Testdaten zu wählen sind und wie mögliche Abweichungen ihrer Ausgaben beurteilt werden sollen. Für eine systematische Antwort dieser Fragen bilden Sicherheitsmodelle die Grundlage, wie im Folgenden gezeigt werden soll.

Neben dynamischen Methoden gibt es auch statische Analysen, die für einen Nachweis der Gleichwertigkeit herangezogen werden können. So ist hier sicherlich das Model Checking zu nennen. Diese Arbeit beschränkt sich auf dynamische Methoden. Es werden verschiedene Kriterien hierfür vorgestellt. In einer späteren Untersuchung sollen weitere, insbesondere auch statische Ansätze verfolgt werden.

1.1 Voraussetzung: Komponentenbasierte Risikoanalyse

Nach (DIN EN 62304) ist die Systementwicklung durch ein geeignetes Risikomanagement (DIN EN ISO 14971) abzusichern. Dazu gehört ein Verfahren zur Erkennung und Minimierung von Einflüssen und Risiken. Etablierte Verfahren hierzu sind Fehlerbaum-, Auswirkungs- und Hazard-Analysen. In ihren ursprünglichen Varianten wurden sie zur Untersuchung von Risiken ganzer Systeme entwickelt, mit Hardware im Fokus.

Für die obige Fragestellung eines zu rechtfertigenden Komponentenaustausches sind sie jedoch unzureichend, da wir sie auf einzelne Komponenten und ihre Risiken anwenden müssen. Betrachten wir dazu ein einfaches System, bestehend aus einer Sensor-, Software- und Aktuatorkomponente. Software altert nicht, Fehlverhalten ist reproduzierbar und ge-

schieht nicht zufällig über der Zeit. Andererseits treten bei der Verarbeitung systematische Rundungs- und Diskretisierungsabweichungen auf, die jedoch einem anderen statistischen Modell folgen. Die Sensoren ihrerseits haben quantifizierbare, beschränkte Wertediskriminierungen und die Aktuatoren physikalische Trägheiten, so dass es hier zu systematischen Messungenauigkeiten und Ansteuerungen stets kommt. Aus dieser Betrachtung folgt, dass eine induktive Wirkungsanalyse von Abweichungen für dieses komponentenbasierte System angemessener ist.

Kaiser et. al [KLM03] entwickelten einen komponentenbasierten Zugang für die Fehlerbaumanalyse. Giese et.al. [GTS04] schlagen einen eng mit der UML verzahnten Weg vor, Hazard-Analyse und Komponentenmodell zu vereinen. Ihre Grundidee ist, dass Fehler in den SW-Komponenten klassifiziert und darüber typisiert werden können. Die Fehlerauswirkungen pflanzen sich dann über getypte Ports in anderen Komponenten fort. Zu weiteren Ansätzen siehe [GKP05].

In dieser Arbeit soll nur ein sehr vereinfachtes, abstraktes Beispiel zur Illustration genutzt werden.

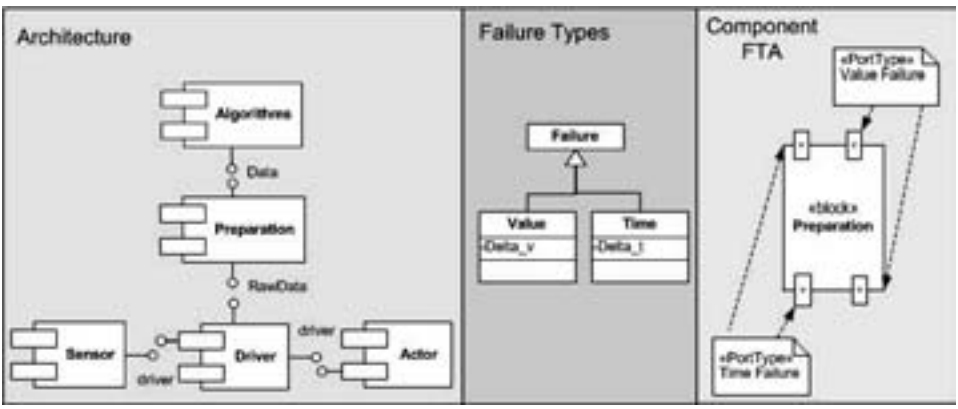

Abbildung 1: Architektur, Fehlertypen und Komponenten FTA

In Abbildung 1 wird ein generischer Aufbau eines eingebetteten Systems als Schichtenmodell mit Treiber/Vorverarbeitung/Algorithmen-Komponenten beschrieben, welches Sensordaten aufnimmt und einen Aktuator ansteuert. Dazu werden im Wesentlichen zwei Fehlertypen betrachtet: zum Einen fehlerhafte Werte, zum Anderen verfrühte oder verspätete Werte. In der dritten Grafik wird dann exemplarisch eine Ursache-Wirkungsanalyse für die Vorverarbeitungskomponente gezeigt, bei der fehlerhafte Werteeingaben und Zeitverzögerungen herein gehen und weitergegeben werden können.

Eine erschöpfende Modellierung ergibt eine detaillierte Sicht auf die möglichen Fehler und ihre Auswirkungen. Dieses Verfahren kann verfeinert werden, in dem man nach den Fehlerabweichungen Delta_v und Delta_t die Fehlertypen weiter differenziert, in unkritische, vertretbare und fehlerhafte Abweichungen. Dies wird nahegelegt, da Sensormessungen, Microcontroller-Berechnungen etc. technisch bedingt nur eine begrenzte Diskriminierung leisten, d.h. Rundungs- und Diskretisierungsabweichungen systematisch mit einzubezie-

hen sind. So kann erfasst werden, dass z.B. eine zeitliche Abweichung von 0.2 s unkritisch sein kann. Addiert sich diese jedoch mit weiteren späteren Verzögerung, verursacht durch eine Berechnung in einer anderen Komponente, so kann die Gesamtabweichung kritisch werden. Anhand einer solchen Verfeinerung des Sicherheitsmodells lassen sich dann Kennzahlen über die Werte-und Zeit-Toleranzen ableiten, die von den einzelnen Komponenten einzuhalten sind.

Als Ergebnis einer solchen Sicherheitsanalyse erhalten wir zweierlei. Zum Einen leitet sich aus der Einordnung des Systems in eine Sicherheitsklasse ab, welche Maßnahmen grundsätzlich zu ihrer Qualitätssicherung zu treffen sind.

Beispiel: Das System gehöre zur Medizin Produkt Klasse III. Dann sind Tests mit folgender Testtiefe durchzuführen:

- 100 % Anforderungsüberdeckung
- 100 % Zweig Überdeckung
- 95 % MCDC
- ...

Daraus erhalten wir, welche Menge von Testdaten mindestens zu beachten ist.

Zum Anderen erhalten wir Vergleichskriterien über erlaubte Abweichungen zwischen den auszutauschenden Komponenten:

Beispiel: Das Ausgabesignal der Komponente K_{neu}, σ_1, darf höchstens um den Wert $\Delta_y > 0$ zum Ausgabewert der Komponente K_{alt} abweichen mit einer maximalen zeitlichen Verzögerung $\Delta_t > 0$.

2 Back-To-Back Tests

Die Leitfrage dieses Artikels ist, unter welchen Umständen ein Austausch zweier Softwarekomponenten sicher durchgeführt werden kann. Dazu wird jetzt untersucht, wie die in der allgemeinen Softwareentwicklung etablierten Techniken von Back-To-Back- oder Regressionstests genutzt werden können. Die allgemeine Idee dieser Tests ist, eine Menge von gleichen Testeingaben auf die zwei Repräsentationen (Back-To-Back-) oder Versionen (Regressions-Tests) zu geben und die jeweiligen Ausgaben gegeneinander zu vergleichen. Damit stellen sich unmittelbar zwei Fragen: Welche Testdaten sollen betrachtet werden (Test Suite) und welche Vergleichskriterien sind heranzuziehen? Ausgehend von einer Risikoanalyse können, wie in 1.1 skizziert, diese Fragen systematisch beantwortet werden. So legt sie beispielsweise fest, ob der der Nachweis korrekter Umsetzung aller Anforderungen durch Tests genügt, ob gegebenenfalls eine bestimmte Testtiefe erreicht werden muss oder andere Aspekte zu verifizieren sind. Ferner zeigt sie, ob komplette Werteverläufe vorzugeben oder bestimmte Eigenschaften von Signalen und ihren Abhängigkeiten gefordert sind. Entsprechend sind dann Wertevergleiche mit Wert- und Zeittoleranzen für

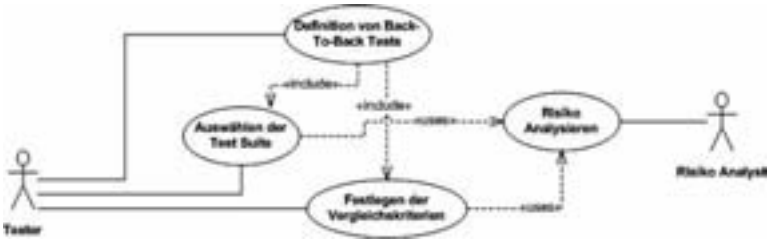

Abbildung 2: Use Cases für einen Back-To-Back Test

einen Vergleich heranzuziehen, siehe 2.2.1, oder Merkmalsbeschreibungen mit geeigneten temporallogischen Formeln, siehe 2.2.2.

2.1 Test Suiten

In diesem Kapitel werden verschiedene Test Suiten charakterisiert, die für einen Nachweis auf Gleichwertigkeit herangezogen werden können. Welche Test Suiten zu nehmen sind, aus welchen Kriterien heraus sie zu entwickeln sind, leitet sich im Allgemeinen aus der Risikoeinstufung des Produktes bzw. der Komponente ab. Nach der Medizinprodukterichtlinie beispielsweise werden sie gemäß ihres Gefährdungspotenzials eingestuft. Daraus leitet sich dann auch ab, in welcher Testtiefe, und damit welcher Test Suite sie abzusichern sind.

2.1.1 Anforderungsbasiert

Nach DIN EN 62340 ist die korrekte Umsetzung von Systemanforderungen durch Tests verfolgbar nachzuweisen. Meist werden die Anforderungen auch auf die einzelnen Komponenten heruntergebrochen. Ausgehend von den Komponentenanforderungen lassen sich dann alle nachweisenden Tests über eine Tracinganalyse identifizieren. Diese bilden dann die anforderungsbasierte Test Suite. Zwei Komponenten sind dann *gleichwertig im Sinne der Anforderungen*, wenn beim Durchlaufen der anforderungsbasierten Test Suite ihre Ausgaben gemäß definiertem Vergleichskriterium übereinstimmen. Gleichwertigkeit in diesem Sinne ist sicherlich eine minimale Forderung.

2.1.2 Strukturbasiert

Weit eingesetzte Kriterien für die Testtiefe von Test Suiten sind Überdeckungsmaße. So werden insbesondere in sehr kritischen Bereichen x % Zweigüberdeckung oder MCDC (Modified Condition/Decision Coverage) verlangt [Lig02]. Dazu muss eine Test Suite erstellt werden, die diese Überdeckung erreicht. Da eine Reimplementierung jedoch andere Strukturen aufweisen kann, ist für den Nachweis der Äquivalenz eine Test Suite zu erstellen, die diese Überdeckungstiefe bezüglich beider Komponenten erreicht. Zwei Kom-

ponenten sind dann *gleichwertig im Sinne der Strukturüberdeckung*, wenn beim Durchlaufen der strukturbasierten Test Suite ihre Ausgaben gemäß definiertem Vergleichskriterium übereinstimmen. Abhängig von der Sicherheitseinschätzung und den Programmiersprachen sind hier die geeigneten Strukturüberdeckungsmaße für den Äquivalenznachweis heranzuziehen, siehe die Diskussionen in [Lig02].

2.1.3 Robustheit

Meist werden die einzelnen Funktionen einer Komponente für sich genommen gut getestet und verglichen. Um nun auch deren Zusammenspiel abzusichern und damit die Robustheit auch gegen unwahrscheinliche Eingabekombinationen zu testen, kann man zufallsgesteuerte Tests durchführen. Da bei reinem, kombinatorischem Zufall jedoch zumeist nur abwegige, uninteressante Situationen gewürfelt werden, ist eine zufallsgesteuerte Testdatengenerierung über Markov-Automaten zu empfehlen, in denen typische Szenarien für die einzelnen Funktionen sowie auch unplausible Werte modelliert wurden. Zur Bestimmung der Testtiefe können kombinatorische Maße genutzt werden.

Die Ausgaben bei stochastischer Ansteuerung können in der Regel nicht mehr systematisch bestimmt werden. Auch können verschiedene Implementierungen hier zu erheblichen Abweichungen führen. Um sie dennoch vergleichen zu können, wird hier ein anderer Weg empfohlen. Ausgehend von den Anforderungen an die Komponente und der Risikoanalyse werden möglichst umfassend allgemeine Eigenschaften gefordert, die einzuhalten sind, siehe Kapitel 2.2.2. Wenn zwei Systeme sich bei stochastischer Ansteuerung und gegebenem Katalog geforderter Eigenschaften und Abhängigkeiten gleich verhalten, so heißen sie *gleichwertig im Sinne der Robustheit*. Dieses Verfahren sollte ergänzend zu den Anforderungs- und Strukturbasierten Ansätzen eingesetzt werden, wenn die Sicherheitsanforderungen sehr hoch sind und zudem vielfältige Funktionalitäten interferieren können.

2.2 Vergleichskriterien

Wie in Kapitel 1.1 skizziert, kann aus aus einer verfeinerten, komponentenbasierten Risikoanalyse ein qualitatives Maß für die erlaubten Abweichungen abgeleitet werden. In diesem Kapitel sollen diese Kriterien etwas genauer untersucht werden. Zunächst wird dabei ein einfaches Kriterium vorgestellt, welches vielfach zur Absicherung der Code Generierung aus Modellen eingesetzt wird.

Zunächst einige Vorbemerkungen. Die Ausgänge eines eingebetteten Systems bilden Werte über der Zeit. Allgemein sollen hier Ausgänge als Zahlenwerte über einen Zeitbereich verstanden werden, im weiteren Signale σ genannt. Es sind verschiedene Zeitmodelle ($\mathbb{Z}$ oder $\mathbb{R}$ z.B.) denkbar. Wir vereinfachen diesen Ansatz und gehen von endlichen Zeitintervallen aus, sogar nur von [0,1]. Dies stellt keine wirkliche Einschränkung dar, siehe [MEv, CSW05]. Im Folgenden werden Kriterien für den Vergleich von Signalen entwickelt, die einen informellen Begriff der Gleichheit operationalisieren. Dabei werden zwei verschiedene Ansätze verfolgt. Im ersten werden die Signaldaten ohne weitere Semantik

mathematisch verglichen. Im zweiten Ansatz hingegen werden die Signalabhängigkeiten und erwarteten Eigenschaften formuliert, die dann die Basis für einen Vergleich geben.

2.2.1 Wertevergleiche

Als reell-wertige Funktionen über die Zeit lassen sich zwei Signale kanonisch über mathematische Normen vergleichen:

Definition 2.1 (Absolute Differenz)
Seien σ_1 und σ_2 Signale mit Werten in $\mathbb{R}$. Dann definieren wir:

$$\|\sigma_1 - \sigma_2\|_\infty := sup_{t\in[0,1]}|\sigma_1(t) - \sigma_2(t)|$$

wobei $|\cdot|$ der absolute Betrag auf $\mathbb{R}$ ist.

Dieses Kriterium lässt sich einfach auf gewichtete Differenzen erweitern, d.h. anstelle des Betrages wird hier die Differenz noch durch geeignete Gewichtsfunktionen modifiziert.

Wenn also für zwei Signale σ_1, σ_2 und vorgegebenem $\epsilon > 0$ gilt: $\|\sigma_1 - \sigma_2\|_\infty < \epsilon$, so werden diese Signale als ähnlich eingestuft.

Dieses Kriterium jedoch ist ungeeignet, auch zeitliche Abweichungen zu berücksichtigen. Eine gängige Erweiterung besteht in der Betrachtung von sogenannten Schlauchumgebungen.

Beispiel: Ein Signal springt zunächst von 0 auf 1 und kurz darauf von 1 auf -1 und zurück auf 0. Das ist signifikant verschieden zu: von 0 auf -1 , kurz darauf von -1 auf 1 und wieder auf 0, auch wenn diese Sprünge sehr dicht beieinander liegen. Legt man jedoch einen Schlauch um beide Signale, können sie leicht beide darin Platz finden. Ein Kriterium, welches die Ähnlichkeit zweier Signale durch solch eine Schlauchumgebung definiert, ist somit nicht geeignet, diese signifikanten Unterschiede zu detektieren, siehe 3.

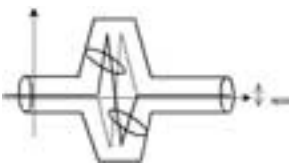

Abbildung 3: Kriterium: Schlauch Differenz $\bigwedge t : \sigma_1(t)$ liegt im ϵ-Schlauch von $\sigma_2(t)$

Es wird deshalb ein anderes Kriterium hier vorgeschlagen. Zwei Ausgänge σ_1, σ_2 heißen (Δ_y, Δ_t) gleich, wenn es (orientierungserhaltene) Zeitreparametrisierungen $\gamma_1, \gamma_2 \in Diff_0([0,1])$ gibt mit

$$\|\sigma_1 \circ \gamma_1(t) - \sigma_2(t)\| < \Delta_y, \quad \forall t \in [0,1]$$
$$\|\gamma_1(t) - t\| < \Delta_t \quad \forall t \in [0,1]$$

und ebenso für $1 \leftrightarrow 2$.

Mathematisch präzisiert:

Definition 2.2
Seien σ_1 und σ_2 Signale mit Werten in $\mathbb{R}$. Dann definieren wir für $\Delta > 0$:

$$d_\Delta^\infty(\sigma_1, \sigma_2) := inf_{U_\Delta(id)} \|\sigma_1 \circ \gamma - \sigma_2\|_\infty$$

wobei $\|\cdot\|_\infty$ die Differenz aus Definition 2.1 ist und $U_\Delta(id) = \{\gamma \in Diff_0([0,1]) |\ \|\gamma - id\|_\infty < \Delta\}$.

Die Existenz und Eindeutigkeit einer solchen Größe lässt sich über mengentheoretische Eigenschaften von $Diff_0([0,1])$ und der vollständigen Ordnung der reellen Zahlen zeigen (Zornsches Lemma).

Zwei Signale σ_1, σ_2 heißen $(\Delta_y, \Delta_t)-$ ähnlich, g.d.w. $d_{\Delta_t}^\infty(\sigma_1, \sigma_2) < \Delta_y$ gilt.

Für die diskretisierten Signale (Zeitmodell $[0,1] \cap \mathbb{Z} \cdot \delta$, δ die Schrittweite) existieren Algorithmen und Implementierungen (Differenzmatrix-Verfahren, Dynamic Adaption), die einen automatischen Vergleich von Ausgängen nach diesen Kriterien unterstützen [MEv, CSW05]. Der Tester gibt für die relevanten Ausgangssignale die Toleranzen in den Werte- und Zeitabweichungen an. Das Werkzeug sucht dann eine geeignete Reparametrisierung und vergleicht anschließend die Signale nach Definition 2.2.

2.2.2 Merkmalsvergleiche

Ein anderer Zugang zu Vergleichskriterien geht über eine qualitative Beschreibung der Ausgänge. Dabei wird auch eine weitere Fragestellung aus dem Fallbeispiel aufgegriffen.

> Wie kann man Anforderungen modellieren ohne die Lösung vorweg zu nehmen? [SPE09]

Diese Frage ist eng verbunden mit einer Beschreibung der zu erwartenden Ausgaben eines Systems. Zur Konstruktion von Eingangssignalen eines Systems gibt es zahlreiche Ansätze. Darin werden die Verläufe durch abschnittsweise Rampen, Splines etc. mit endlich vielen Stützstellen beschrieben. Viele erprobte Testsysteme setzen hier auf (CTE, MTest: [GK93, Con04]). Jedoch reichen diese Ansätze nicht allzu weit, will man die Verläufe der Ausgangssignale ebenso erfassen. Diese folgen nicht unbedingt elementaren Funktionsverläufen. Erwartete Nulldurchgänge, Signalanstiege, Extrema und andere markante Stellen sind relevant.

In [GW07, Loo08] werden Konzepte vorgestellt und Algorithmen skizziert, wie ein deskriptiver Ansatz realisiert werden kann. Im Gegensatz zum konstruktiven Ansatz, der häufig von einer vorgegeben Zeiteinteilung ausgeht, folgt man hier dem erwarteten Werteverlauf. In einem ersten Schritt wird der erwartete qualitative Verlauf eines Signales beschrieben, also, z.B. eine Liste der Art: Erst Nulldurchgang, dann steiler Anstieg (Steigung

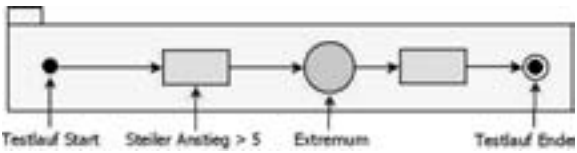

Abbildung 4: Verlaufsdiagramm zur Beschreibung eines einfachen Signalverlaufes

>1), Maximum,.... Dazu wird eine grafische Notation vorgeschlagen, in der formalisiert ein solcher qualitativer Verlauf erfasst werden kann, siehe Abbildung 4.

Das Eintreten so spezifizierter Merkmale wie z.B. eine Flagänderung bildet ein Ereignis. Werden nun verschiedene Signale mit ihren Merkmalsbeschreibungen betrachtet, lassen sich über die in den Verlaufsdiagrammen spezifizierten Ereignisse temporallogische Abhängigkeiten fordern. Insbesondere können in diesen Formeln Zeittoleranzen kodiert werden wie: Synchron bis auf Δ_t-Abweichung. Auf diese Weise können geforderte Signalabhängigkeiten spezifiziert und in Testdurchläufen algorithmisch überprüft werden. Zwei Signalausgaben sind dann äquivalent, wenn sie definierten Verlaufsdiagrammen und temporallogischer Abhängigkeiten genügen.

Dieser Ansatz kann dazu verwendet werden, Signalmuster zu definieren und Ausgänge auf allgemein verbindliche Abhängigkeiten zu überprüfen. Dies ist besonders für stochastische Ansteuerungen wichtig, bei denen die erwarteten Ausgangswerte nicht mit vertretbarem Aufwand vorweg bestimmt werden können. In einem Forschungsprojekt aus dem Automotive-Bereich wurde eine Variante dieses Vorgehens prototypisch umgesetzt und bereits sehr erfolgreich eingesetzt.

3 Vorgehen

Es wird eine Softwarekomponente reimplementiert, sei es, weil neue Hardware eingesetzt werden soll, der ursprüngliche Code nicht mehr wartbar war, kein gewarteter Compiler mehr existiert oder neue Programmierparadigma berücksichtigt werden sollten. Der Entwicklungsprozess, auch für die neue Komponente, ist bereits konform zu den gültigen Normen.

Im Folgenden gehen wir von einer erschöpfenden Risikoanalyse aus, die insbesondere auf Komponentenebene verfeinert wurde, siehe Abschnitt 1. Aus dieser Analyse lassen sich nun in Absprache mit der Zertifizierungsbehörde Kriterien festlegen, unter welchen Bedingungen die Komponenten als gleichwertig im Sinne eines kontrollierten Austausches gelten können. Dazu können statische Analysen ebenso gehören, wie dynamische Testtechniken, z.B. Back-To-Back-Tests. In Abbildung 5 wird exemplarisch und vereinfacht gezeigt, wie ein Vorgehen im Falle von Back-To-Back-Tests aussehen könnte.

Ausgehend von der komponentenbasierten Risikoanalyse werden mit der Zertifizierungsbehörde Kriterien abgestimmt, Test Suite und Vergleichskriterien, die dann vom Hersteller durchgeführt werden. Falls keine signifikanten Abweichungen auftreten, können die bei-

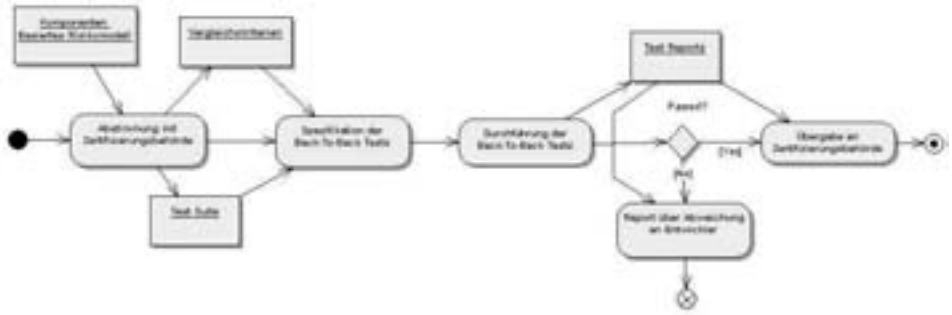

Abbildung 5: Vorgehen zur Rezertifizierung

den Komponenten als gleichwertig angesehen werden und ein Austausch ist abgesichert.

In dieser Arbeit konnte nur eine grobe Skizze gegeben werden. Viele Ideen, so insbesondere das Testen auf Robustheit über Markov-Automaten und ihre Auswertung über Merkmalsbeschreibungen, wie auch die Verwendung statischer Analyse-Kriterien werden zurzeit im Rahmen des Forschungsprojektes SPES 2020 weiter bearbeitet.

Literatur

[Bal96] H. Balzert. *Lehrbuch der Software-Technik, Bd. 1: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, 1996.

[BMR+98] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad und M. Stal. *Pattern-orientierte Software-Architektur*. Addison-Wesley, Bonn, 1998.

[Con04] M. Conrad. *Modell-basierter Test eingebetteter Software im Automobil*. Dissertation, TU-Berlin, 2004.

[CSW05] M. Conrad, S. Sadeghipour und H.-W. Wiesbrock. Automatic Evaluation of ECU Software Tests. In *SAE World Congress*, Detroit (US), 2005.

[GHJV95] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Design Patterns: Abstraction and Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GK93] M. Grochtmann und K.Grimm. Classification Trees For Partition testing. In *Software testing, Verification & Reliability*, Jgg. 3 , Number 2, Seite 63–82. Wiley, 1993.

[GKP05] L. Grunske, B. Kaiser und Y. Papadopoulos. Model-Driven Safety Evaluation with State-Event-Based Component Failure Annotations. In *CBSE*, Seiten 33–48, 2005.

[GTS04] H. Giese, M. Tichy und D. Schilling. Compositional Hazard Analysis of UML Component and Deployment Models. In *SAFECOMP*, Seiten 166–179, 2004.

[GW07] C. Gips und H.-W. Wiesbrock. Proposal for an Automatic Evaluation of ECU Output Signals. In *Dagstuhl-Workshop MBEES 2007*, Braunschweig, 2007.

[KLM03] B. Kaiser, P. Liggesmeyer und O. Mäckel. A New Component Concept for Fault Trees. In *SCS*, Seiten 37–46, 2003.

[Lig02] P. Liggesmeyer. *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Springer, 2002.

[Loo08] M. Loose. *Entwicklung eines graphischen Editors und zugehöriger Algorithmen zur werkzeugunterstützten Überprüfung von Signalabhängigkeiten*. Diplomarbeit, HU Berlin, 2008.

[MEv] MEval. `www.itpower.de/12-0-Tool-Produkte.html`.

[SPE09] SPES 2020, Teilprojekt Medizintechnik, Berlin Heart. Beschreibung der Fallstudie im Anwendungsgebiet Medizintechnik, November 2009.