Forschungsbericht
Institut für Automatisierungstechnik und Softwaresysteme

Hrsg.: Prof. Dr.-Ing. Dr. h. c. Michael Weyrich

Andreas Zeller

Absicherung von verteilten Automatisierungssystemen nach Änderungen der Steuerungssoftware

Modellkomposition zur Nutzung der funktionalen Verifikation

Band 2/2019

Universität Stuttgart

IAS-Forschungsberichte

Absicherung von verteilten Automatisierungssystemen nach Änderungen der Steuerungssoftware

–

Modellkomposition zur Nutzung der funktionalen Verifikation

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Andreas Zeller
aus Leonberg

Hauptberichter:	Prof. Dr.-Ing. Dr. h. c. Michael Weyrich
Mitberichter:	Prof. Dr.-Ing. Christian Diedrich
Tag der Einreichung:	04.04.2019
Tag der mündlichen Prüfung:	09.10.2019

Institut für Automatisierungstechnik und Softwaresysteme
der Universität Stuttgart

2019

IAS-Forschungsberichte

Band 2/2019

Andreas Zeller

Absicherung von verteilten Automatisierungs-systemen nach Änderungen der Steuerungssoftware

Modellkomposition zur Nutzung der funktionalen Verifikation

D 93 (Diss. Universität Stuttgart)

Shaker Verlag
Düren 2019

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: Stuttgart, Univ., Diss., 2019

ISBN 978-3-8440-7039-2
ISSN 1610-4781

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Telefon: 02421 / 99 0 11 - 0 • Telefax: 02421 / 99 0 11 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungstechnik und Softwaresysteme (IAS) der Universität Stuttgart.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Dr. h.c. Michael Weyrich für die Übernahme des Hauptberichts, die Betreuung dieser wissenschaftlichen Arbeit sowie die zahlreichen konstruktiven Anregungen und Diskussionen, welche zum Entstehen dieser Arbeit beigetragen haben.

Herrn Prof. Dr.-Ing. Christian Diedrich danke ich für das entgegengebrachte Interesse an meiner Arbeit und die Übernahme des Mitberichts.

Allen Kolleginnen und Kollegen am IAS danke ich für die motivierende Zusammenarbeit und die gegenseitige Unterstützung. Insbesondere danke ich Dr.-Ing. Nasser Jazdi für die wertvollen Impulse, die Hilfsbereitschaft und die kritische Durchsicht der Dissertationsschrift.

Dank gilt auch den Studierenden, welche mit ihren Master-, Bachelor- und Forschungsarbeit einen wesentlichen Anteil an der prototypischen Realisierung des Konzepts haben.

Schließlich möchte ich mich bei meiner Familie und meinen Freunden für den langjährigen Rückhalt und die Motivation bedanken. Besonders danke ich meiner Schwester Dr. phil. Claudia Gruhn für die sprachliche Durchsicht der Dissertationsschrift und meiner Freundin Sina Brenner für das vertrauensvolle Verhältnis, welches mir viel Kraft, Antrieb und Freude gibt.

Stuttgart, im Oktober 2019

Andreas Zeller

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ARIS	**Ar**chitektur integrierter **I**nformations**s**ysteme
ATS	**A**utoma**t**isierungs**s**ystem
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage
BPMN	**B**usiness **P**rocess **M**odel and **N**otation
CTL	**C**omputation **T**ree **L**ogic
FMEA	**F**ehler**m**öglichkeits- und **-e**influssanalyse
GRAFCET	**Gra**phe **F**onctionnel de **C**ommande **E**tapes/**T**ransitions
JDBC	**J**ava **Data**base **C**onnectivity
JSON	**J**ava**s**cript **O**bject **N**otation
JSP	**J**ava**S**erver **P**ages
M2M	**M**achine-**to**-**M**achine
NCES	**N**et **C**ondition/**E**vent **S**ystems
LAN	**L**ocal **A**rea **N**etwork
LTL	**L**inear **T**emporal **L**ogic
OASIS	**O**rganization for the **A**dvancement of **S**tructured **I**nformation **S**tandards
OMG	**O**bject **M**anagement **G**roup
OPC UA	**O**pen **P**latform **C**ommunications **U**nified **A**rchitecture
PIPN	**P**rozessbezogen **i**nterpretierte **P**etri-**N**etze
PNML	**P**etri **N**et **M**arkup **L**anguage
QS	**Q**ualitätssicherung
R-TNCES	**R**econfigurable **T**imed **N**et **C**ondition/**E**vent **S**ystem
SFC	**S**equential **F**unction **C**hart

SIPN	**S**teuerungstechnisch **i**nterpretierte **P**etri-**N**etze
SPENAT	**S**icheres **Pe**tri-**N**etz mit **At**tributen
SysML	**Sys**tems **M**odeling **L**anguage
SOA	**S**ervice**o**rientierte **A**rchitektur
TestIAS	**Test**system am **I**nstitut für **A**utomatisierungstechnik und **S**oftwaresysteme
TTCN-3	**T**esting and **T**est **C**ontrol **N**otation
TCP/IP	**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol
UML	**U**nified **M**odeling **L**anguage
UML-Marte	**U**nified **M**odeling **L**anguage – **M**odeling and **A**nalysis of **R**eal **T**ime and **E**mbedded systems
UML-PA	**U**nified **M**odeling **L**anguage – **P**lant **A**utomation
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
XML	E**x**tensible **M**arkup **L**anguage

Begriffsverzeichnis

Abhängigkeitsmodell: Darstellung von Abhängigkeiten zwischen Komponenten eines Systems.

Anlagenbetreiber: Person, die die Gesamtverantwortung für den sicheren Betrieb der Anlage trägt und die Regeln und Randbedingungen der Organisation vorgibt (nach [1]).

Asynchrone Kommunikation: Kommunikationsmodus, bei dem Senden und Empfangen von Daten zeitlich versetzt auftreten können. Dies muss nicht dazu führen, dass der Sender blockiert ist, bis er eine Antwort erhält.

Auswirkungsanalyse: Die Untersuchung und Auswertung der Auswirkungen einer Änderung an einer Komponente auf die Gültigkeit funktionaler Anforderungen anderer Komponenten.

Automatisierungssystem: Ein System mit dem Ziel der Automatisierung technischer Prozesse [2].

Cyber-physisches Produktionssystem: Ist ein Cyber-physisches System, das in der Produktion eingesetzt wird. Cyber-physisch beschreibt dabei, dass reale (physische) Objekte und Prozesse mit informationsverarbeitenden (virtuellen) Objekten und Prozessen über offene, teilweise globale Informationsnetze verbunden sind [3].

Digitaler Zwilling: Digitale Repräsentanz eines Objekts aus der realen Welt. Der Digitale Zwilling enthält Informationen des Objekts aus dessen gesamten Lebenszyklus. Diese Informationen können unter anderem Daten, Modelle und Algorithmen sein.

DOT: Ein textbasiertes Dateiformat zur Beschreibung der visuellen Darstellung von Modellen.

Fehlermöglichkeits- und Einfluss-Analyse: Ein systematischer Ansatz zur Risikoidentifikation sowie zur Analyse möglicher Fehler(aus)wirkungen und zur Vermeidung von Fehlern [4].

Flexible Produktionssysteme: Integriertes System rechnergesteuerter Maschinenmodule und Materialtransporteinrichtungen für die automatisierte, zufallsgesteuerte Verarbeitung von Teilen. Das Ziel ist es, vordefinierte Produktfamilien, die sich ändern können, kosteneffizient mit geforderter Qualität und Quantität zu produzieren [5].

Horizontale Integration: Integration innerhalb einer funktionalen/organisatorischen Hierarchieebene über Systemgrenzen hinweg [3].

Industrie 4.0: Zukunftsprojekt zur Förderung der Digitalisierung in der industriellen Produktion.

Komponente: Eine Software-Komponente ist ein Software-Baustein, der konform zu einem Komponentenmodell ist und nach einem Composition-Standard ohne Änderungen mit anderen Komponenten verknüpft, ausgeführt und wiederverwendet werden kann [6].

Komposition: Zusammenfügen mehrerer Modelle zu einem Gesamtmodell über definierte Schnittstellen.

Lebenszyklus: Folge von Prozessen, die ein Gegenstand oder eine Komponente von der Entstehung bis zum Vergehen durchläuft.

Lose Kopplung: Ziel der losen Kopplung ist die Reduktion der Abhängigkeit zwischen Software-Komponenten. Die Komponenten binden sich zur Laufzeit und besitzen einen azyklischen Kommunikationsstil.

Modell: Vereinfachtes Abbild der Realität unter einer bestimmten Sicht [7].

Orchestrierung von Services: Flexibles Verbinden von einzelnen Diensten für einen definierten Zweck. Anmerkung: Dies kann während der Planungsphase und/oder zur Laufzeit erfolgen [8].

Over-the-air (OTA) Updates: Aufspielen neuer Software-Versionen über drahtlose Kommunikationsnetzwerke.

Platzhalter: Eine rudimentäre Modellierung einer Komponente oder eines Systems.

Redundanz: Verwendung mehrerer Elemente oder Systeme zur Durchführung der gleichen Funktion [9].

Regressionstest: Erneutes Testen eines bereits getesteten Programms bzw. einer Teilfunktionalität nach deren Änderung. Ziel ist es, nachzuweisen, dass durch die vorgenommenen Änderungen keine Fehlerzustände eingebaut oder bisher maskierte Fehlerzustände freigelegt wurden. Anmerkung: Ein Regressionstest wird durchgeführt, wenn die Software oder ihre Umgebung verändert wurde [10].

Rekonfigurierbare Produktionssysteme: Rekonfigurierbare Produktionssysteme sind so konzipiert, dass eine schnelle Strukturänderung möglich ist, um zügige Anpassungen der Produktionskapazität und Funktionalität gemäß geänderter Marktanforderungen durchzuführen [5].

Safety: Abwesenheit von Gefahren die von einem System auf die Umwelt ausgehen.

Schnittstelle: Definierte Verbindungsstelle einer Funktionseinheit, über die diese mit anderen Funktionseinheiten verbunden werden kann [3].

Service: (deutsch: Dienst) Abgegrenzter Funktionsumfang, der von einer Entität oder Organisation über Schnittstellen angeboten wird [11].

Serviceorientierung: Paradigma, welches das einfache Austauschen, Hinzufügen und Entfernen von lose gekoppelten Services ermöglicht [8].

Serviceorientierte Architektur: Paradigma zur Nutzung und Organisation von Services in verteilten informationstechnischen Systemen.

Software-Änderung: Modifikation des Programmcodes einer Software.

Steuerungssystem: Einheit zur Beeinflussung eines technischen Prozesses. Dabei werden ausgehend von Sensordaten und dem inneren Zustand des Steuerungssystems Aktorsignale berechnet.

Strukturmodell: Beschreibung, um die Struktur von Systemen abstrakt darzustellen.

System: Menge von Komponenten, die in Beziehung stehen [3].

Technischer Prozess: Gesamtheit aufeinander einwirkender Vorgänge eines technischen Systems, durch welche Materie, Energie oder Information umgeformt, transportiert oder gespeichert werden und dessen physikalische Größen mit technischen Mitteln erfasst und beeinflusst werden können [7].

Validierung: Überprüfung, ob eine Software den Praxisanforderungen entspricht. Sie stellt fest, ob sie dem jeweiligen Einsatzzweck genügt.

Verhaltensmodell: Beschreibung, um das Verhalten von Systemen abstrakt darzustellen.

Verifikation: Überprüfung, ob eine Software ihrer Spezifikation entspricht. Zur Feststellung, ob nach Erstellung der Anforderungen Fehler gemacht wurden.

Verteiltes System: „Ein verteiltes System ist definiert durch eine Menge von Funktionen oder Komponenten, die in Beziehung zueinander stehen (Client-Server-Beziehung) und eine Funktion erbringen, die nicht erbracht werden kann durch die Komponenten alleine“ [12].

Vertikale Integration: Integration innerhalb eines Systems über funktionale/organisatorische Hierarchie-Ebenen hinweg [3].

Wohldefinierte Schnittstelle: Syntaktisch wie semantisch definierte Schnittstelle.

Zusammenfassung

Wirtschaftliche und technologische Treiber führen dazu, dass sich Steuerungen von Automatisierungssystemen zunehmend zu komplexen Softwaresystemen wandeln. Dabei ermöglicht Software die Realisierung komplexer Steuerungsaufgaben und, aufgrund der leichten Änderbarkeit, ein wandlungsfähiges Steuerungssystem. Jede Änderung birgt aber auch das Risiko, dass dabei eingebrachte Fehler zu einem Fehlverhalten oder dem Ausfall eines Systems führen. Insbesondere bei sicherheitskritischen Anlagen ist das Testen nach Änderung der Steuerungssoftware daher unabdingbar. Eine besondere Herausforderung stellt dabei Testen von verteilten Software-Systemen dar, da oftmals komplexe Abhängigkeitsbeziehungen existieren. Da Funktionsänderungen an Steuerungssystemen zunehmend in der Betriebsphase notwendig werden, stellt dies hohe Anforderungen an Anlagenbetreiber. Da für Anlagenbetreiber in der Vergangenheit hauptsächlich prozesstechnische und maschinenbauliche Fragestellungen relevant waren, besitzen sie oftmals weniger Erfahrung im Bereich des Softwaretests.

Zur Unterstützung von Anlagenbetreibern bei der Absicherung von softwarebasierten Änderungen an Steuerungssystemen wird in der vorliegenden Arbeit ein Konzept für einen strukturierten und automatisierten Absicherungsprozess vorgestellt. Es zielt darauf ab, durch eine automatisierte Auswirkungsanalyse und Generierung formaler Verhaltensmodelle, formale Verifikationsmethoden für Anlagenbetreiber nutzbar zu machen.

Die automatisierte Generierung der zur Verifikation notwendigen Verhaltensmodelle basiert auf einem modularen Modellierungsansatz und Modelloperationen. Ausgehend von der Software-Änderung einer Steuerungskomponente werden, anhand einer Auswirkungsanalyse, die funktionalen Anforderungen identifiziert, deren Gültigkeit nach der Änderung nicht mehr gewährleistet ist. Für die betroffenen funktionalen Anforderungen wird anhand verschiedener Modelloperationen das zur Verifikation notwendige Verhaltensmodell generiert. Über formale Verifikationsmethoden wird anschließend überprüft, ob das generierte Verhaltensmodelle den spezifizierten funktionalen Anforderungen genügt.

Das Konzept wurde in Form des Testgeräts „TestIAS“ umgesetzt. Im Rahmen einer empirischen Evaluierung wurden an einem verteilten, serviceorientierten Automatisierungssystem neun Änderungsszenarien durchgeführt, welche durch TestIAS überprüft wurden. Damit konnte gezeigt werden, dass sich mithilfe des entwickelten Konzepts ein Absicherungsprozess umsetzen lässt, der für Anlagenbetreiber automatisiert durchführbar ist. Als Ergebnis dieses Prozesses stehen die Aussagen, welche Teilsysteme von einer Funktionsänderung betroffen sind und ob durch die Funktionsänderung funktionale Anforderungen verletzt wurden. Sind funktionale Anforderungen verletzt, wird der Anlagenbetreiber bei Eingrenzung der Fehlerursache unterstützt. Dazu benötigt der Anlagenbetreiber keinerlei Kenntnis über die Abhängigkeiten innerhalb des Automatisierungssystems. Dies geschieht, indem durch eine automatisierte Modellgenerierung die Vorteile funktionaler Verifikationsmethoden für Anlagenbetreiber nutzbar gemacht werden.

Abstract

Economic and technological drivers mean that control systems of automation systems are increasingly turning into complex software systems. Software enables the realization of complex control tasks and, due to its easy modifiability, a versatile control system. However, every modification also entails the risk that errors introduced in the modification process could lead to malfunction or failure of a system. Testing after modifying the control software is therefore indispensable, especially for safety-critical systems. Testing distributed software systems poses a particular challenge, as complex dependency relationships often exist. Since functional modifications to control systems are increasingly necessary in the operating phase, high demands are placed on production line operators. In the past, production line operators were mainly concerned with process engineering and mechanical engineering issues, so they often have less experience in software testing.

This paper presents a concept for a structured and automated verification process, to support production line operators in verifying software-based modifications to control systems. It aims at making formal verification methods usable for plant operators by an automated impact analysis and generation of formal behavior models.

The automated generation of the behavior models that are necessary for verification is based on a modular modeling approach and model operations. Starting from the software modification of a control component, the functional requirements of which the validity is no longer guaranteed after the modification are identified. This is realized on the basis of an impact analysis. The behavioral model that is necessary for verification of an affected functional requirement is generated by using various model operations. Then formal verification methods are used to check whether the generated behavior models meet the specified functional requirements.

The concept was implemented by a test device named "TestIAS". Within the framework of an empirical evaluation, nine modification scenarios were carried out on a distributed, service-oriented automation system. Those modifications were verified by TestIAS. This showed that the developed concept can be used to implement a verification process which can be automated for production line operators. As a result of this process, information is given on which subsystems are affected by a functional modification and whether functional requirements are violated by the functional modification. If functional requirements are violated, the plant operator is supported in localizing the cause of the error. For this purpose, the plant operator does not require any knowledge of the dependencies within the automation system. This is done by making the advantages of functional verification methods available to production line operators through automated model generation.