TECHNISCHE UNIVERSITÄT DRESDEN

PROCESS CONTROL SYSTEMS PROCESS SYSTEMS ENGINEERING

School of Engineering Chair of Process Control Systems & Process Systems Engineering Group

Studienarbeit

zum Thema

Teststrategien für Software- und Hardwarekompatibilität in industriellen Steuerungen

vorgelegt von Marcus Rothhaupt
im Studiengang Informationssystemtechnik, Jg. 2018

Betreuer: Dipl.-Ing. Lucas Vogt
Dipl.-Ing. Anselm Klose
Verantwortlicher Hochschullehrer: Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung: 17.10.2022

TECHNISCHE UNIVERSITÄT DRESDEN

PROCESS CONTROL SYSTEMS PROCESS SYSTEMS ENGINEERING

School of Engineering Chair of Process Control Systems & Process Systems Engineering Group

Aufgabenstellung für die Studienarbeit
für
Marcus Rothhaupt, Matr.Nr. 4764547, Informationssystemtechnik 2018

Teststrategien für Software- und Hardwarekompatibilität in industriellen Steuerungen

Kontext

Mit Konzepten wie Simultaneous Engineering werden die Entwicklung von Software- und Hardwarekomponenten für industrielle Anlagen zunehmend voneinander entkoppelt. Dennoch muss bei der Inbetriebnahme einer Anlage die Kompatibilität der verwendeten Software und Hardware stets sichergestellt werden können. Ziel dieser Arbeit ist es daher eine Teststrategie zu entwickeln, welche eine automatisierte Kompatibilitätsprüfung während eines Neustarts bzw. während des Aufspielens von Software auf eine industrielle Steuerung ermöglicht.

Wissenschaftliche Fragestellungen

- Welche wesentlichen Teststrategien, beispielsweise aus der Virtuellen Inbetriebnahme oder der Cyber Security gibt es, und wie lassen sie sich auf Software-Upload bzw. Software-Neustart übertragen, um die Kompatibilität von Software und Hardware sicherzustellen?
- Welche wesentlichen Eigenschaften bzw. Informationen müssen Software- und Hardwarekomponenten aufweisen, um einen Kompatibilitätstest zu ermöglichen?
- Wie kann ein Konzept aussehen, welches angemessene Reaktionen auf gefundene Inkompatibilitäten bzw. Fehler ermöglicht?

Lastenheft

1. Literaturrecherche und begründete Auswahl der Forschungsmethodik zur Bearbeitung der Fragestellungen. Das schriftliche Ergebnis dieses Arbeitspakets dient als Meilenstein
2. Zielgerichtete Beantwortung der Fragestellung durch systematische Anwendung der ausgewählten Forschungsmethodik
3. Kritische abschließende Bewertung der gewählten Arbeitsweise und der Forschungsergebnisse

Die wissenschaftliche Arbeit ist gemäß der Richtlinie des Instituts für Automatisierungstechnik durchzuführen. Eignung und Qualität von erstellter Software sind durch Komponenten-, Integrations- und Systemtests nachzuweisen.

Betreuer:	Dipl.-Ing. Lucas Vogt, Dipl.-Ing. Anselm Klose
1.Prüfer:	Prof. Dr.-Ing. habil. Urbas
Datum Arbeitsbeginn:	02.05.2022
Einzureichen am:	17.10.2022

Prof. Dr.-Ing. habil. Leon Urbas
Prüfer

Marcus Rothhaupt

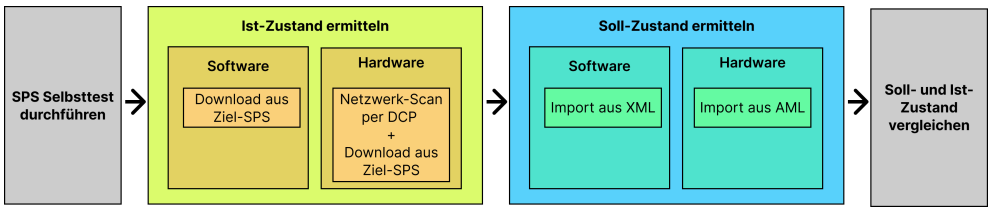
TECHNISCHE UNIVERSITÄT DRESDEN

PROCESS CONTROL SYSTEMS | PROCESS SYSTEMS ENGINEERING

School of Engineering Chair of Process Control Systems & Process Systems Engineering Group

Teststrategien für Software- und Hardwarekompatibilität in industriellen Steuerungen

Massenanpassung, kleine Losgrößen, hohe Variabilität der Produkttypen und ein sich während des Lebenszyklus einer industriellen Anlage änderndes Produktportfolio sind aktuelle Trends der Industrie. Durch eine zunehmende Entkopplung der Entwicklung von Software- und Hardwarekomponenten im industriellen Kontext, entstehen immer häufiger Kompatibilitätsprobleme innerhalb von industriellen Steuerungen. In dieser Arbeit wird mittels Literaturrecherche und angewandter Forschung ein Strategiekonzept zur Kompatibilitätsprüfung hergeleitet und diskutiert. Dieses vierphasige Konzept ermittelt Inkompatibilitäten zwischen Software- und Hardwarekomponenten im Umfeld von industriellen Steuerungen und ermöglicht Testingenieuren das frühzeitige Erkennen von Problemen. Durch eine automatische Durchführung der Kompatibilitätsprüfung auf einem externen Industrie PC kann die Kompatibilitätsprüfung sowohl beim Aufspielen neuer Software auf die industrielle Steuerung als auch beim Neustart der Steuerung ablaufen. Somit werden Änderungen an den Komponenten stetig erkannt und Inkompatibilitäten vermieden. Weiterhin kann durch die frühzeitige Erkennung sichergestellt werden, dass eine Anlage dauerhaft lauffähig bleibt. Anhand einer Diskussion werden Mittel festgestellt, um die Robustheit und Anwendbarkeit des vorgestellten Konzeptes zusätzlich zu festigen.



Betreuer:	Dipl.-Ing. Lucas Vogt
	Dipl.-Ing. Anselm Klose
Hochschullehrer:	Prof. Dr.-Ing. habil. Leon Urbas
Tag der Einreichung:	17.10.2022

STUDIENARBEIT

Bearbeiter: Marcus Rothhaupt

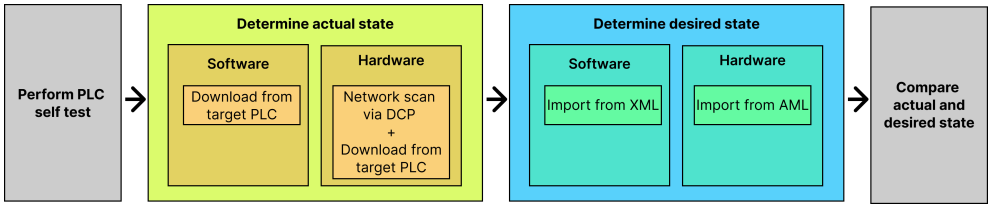
TECHNISCHE UNIVERSITÄT DRESDEN

PROCESS CONTROL SYSTEMS | PROCESS SYSTEMS ENGINEERING

School of Engineering Chair of Process Control Systems & Process Systems Engineering Group

Strategies for software and hardware compatibility testing in industrial controllers

Mass customization, small batch sizes, high variability of product types and a changing product portfolio during the life cycle of an industrial plant are current trends in the industry. Due to an increasing decoupling of the development of software and hardware components in an industrial context, compatibility problems within industrial control systems arise more and more frequently. In this thesis, a strategy concept for compatibility testing is derived and discussed by means of literature review and applied research. This 4-phased strategy concept identifies incompatibilities between software and hardware components in the industrial control environment and enables test engineers to detect problems at an early stage. By automating the compatibility test on an external I-PC, the test can be run both when new software is installed on the industrial controller and when the controller is restarted. Thus, changes to the components are constantly detected and incompatibilities are avoided. Furthermore, early incompatibility detection can ensure that a system remains permanently operational. Based on a discussion, additionally strategies are identified to consolidate the robustness and applicability of the presented concept.



Tutor:	Dipl.-Ing. Lucas Vogt
	Dipl.-Ing. Anselm Klose
Supervisor:	Prof. Dr.-Ing. habil. Leon Urbas
Day of Submission:	17.10.2022

STUDENT RESEARCH THESIS

Author: Marcus Rothhaupt

Inhaltsverzeichnis

1 Motivation **1**

1.1 Aufgabenanalyse 3
1.1.1 Forschungsfragen und Teilaufgaben 3
1.1.2 Aufgabenkomplexe 4
1.1.3 Eingrenzung der Aufgabenstellung 5
1.1.4 Ziel der Arbeit 6
1.1.5 Festsetzung von Formulierungen 6

2 Einführung und Stand der Technik **7**

2.1 VIBN von industriellen Anlagen 7
2.1.1 Teststrategien aus der VIBN 9
2.1.1.1 Model-in-the-Loop 9
2.1.1.2 Software-in-the-Loop 9
2.1.1.3 Hardware-in-the-Loop 10
2.1.1.4 Konklusion und Forschungsbestrebungen . . 11
2.2 CS in industriellen Anlagen 12
2.2.1 Sicherheitsziel 13
2.2.2 Teststrategien aus der CS 13
2.2.2.1 Signaturbasierte Erkennung 14
2.2.2.2 Anomaliebasierte Erkennung 14
2.2.2.3 Konklusion und Forschungsbestrebungen . . 16
2.3 Interoperabilität als Kompatibilitätsmaß 16
2.4 Testautomatisierung und Test Case Generierung 17
2.5 Allgemeine Softwareteststrategien 17
2.5.1 Modellbasiertes Testen 17
2.5.2 Funktionale Tests 18
2.6 Allgemeine Hardware Teststrategien 19
2.6.1 Modellbasiertes Testen 19
2.6.2 Manuelles Testen 19
2.7 Interoperabilität in industriellen Anlagen 20
2.7.1 Definitionen der Interoperabilität 20
2.7.2 Herausforderungen der Interoperabilität 22

2.7.3 Implementierung von Interoperabilität 22
2.7.3.1 Syntaktische Interoperabilität 23
2.7.3.2 Semantische Interoperabilität 23
2.7.4 Vertikale Integration 24
2.7.5 Horizontale Integration 25

3 Anforderungsanalyse **27**
3.1 Adaption von Strategien der VIBN und CS 27
3.2 Anforderungen 28
3.2.1 Anforderungen an die Kompatibilitätsprüfung 28
3.2.2 Anforderungen an die Hardwarekomponenten 29
3.2.3 Anforderungen an die Softwarekomponenten 29

4 Konzept **30**
4.1 Komponenten des Teststrategiekonzeptes 30
4.1.1 SPS Selbsttest 32
4.1.2 Export & Import des Soll-Zustandes 32
4.1.3 Ermittlung des Ist-Zustandes 35
4.1.4 Vergleich des Soll- & Ist-Zustandes 35
4.2 Fehlerdetektionstabellen 36
4.3 Reaktionen auf Inkompatibilitäten 38

5 Evaluation **39**
5.1 Methodik und Evaluationskriterien 39
5.2 Anwendungsbeispiel 39
5.3 Referenzsystem für Evaluation 41
5.4 Durchführung Evaluation 41
5.5 Erfüllung der Anforderungen an die Kompatibilitätsprüfung 46

6 Diskussion **48**
6.1 Beantwortung der Forschungsfragen 48
6.2 Diskussion zur Forschungsmethodik 48
6.3 Bewertung des Konzeptes 49

7 Zusammenfassung und Ausblick **50**
7.1 Zusammenfassung 50
7.2 Ausblick und weitere Forschungsarbeit 51

Literaturverzeichnis **52**

Abbildungsverzeichnis

1.1 Änderung des Lebenszyklus mit VIBN nach (Zeller, 2019) . 2

2.1 Engineering mit und ohne VIBN (Lechler et al., 2019) 8
2.2 MiL, SiL und HiL Simulationsübersicht von (Hill et al., 2021) 10
2.3 Teststrategien der VIBN (Lechler et al., 2019) 12
2.4 Testaktivitäten im modellbasierten Testprozess (Magnus, Russ und Krause, 2016) 18
2.5 V-Modell aus der VDI/VDE 2206 (Gräßler et al., 2021) . . . 19

4.1 Ablauf der Kompatibilitätsprüfung 31
4.2 Ermittlung des Soll- und Ist-Zustandes beim Neustart 33
4.3 Ermittlung des Soll- und Ist-Zustandes beim Software Update 34
4.4 Ablauf des DCP-Scans, nach (PROFINET University, 2018) 36

5.1 Modell eines Steuerungsystems von Sartorius mit Software SPS 40
5.2 Modell eines Steuerungsystems von Sartorius mit Hardware SPS 41
5.3 Software Ist-Zustand der SPS ermitteln 43
5.4 TiaExportBlocks nach dem Erfolgreichen Export 44
5.5 Hardware Ist-Zustand der SPS zu AML exportieren 44

Tabellenverzeichnis

4.1 Fehlerdetektionstabelle für SPS Selbsttest 37
4.2 Fehlerdetektionstabelle für Import & Export des Soll-Zustandes 37
4.3 Fehlerdetektionstabelle für Ermittlung des Software Ist-Zustandes 37
4.4 Fehlerdetektionstabelle für Ermittlung des Hardware Ist-Zustandes 37
4.5 Fehlerdetektionstabelle für Vergleich Soll und Ist-Zustand . . 38

Abkürzungs- und Symbolverzeichnis

AML	Automation Markup Language
API	Application Programming Interface
CAEX	Computer Aided Engineering Exchange
CIA	Confidentiality, Integrity, Availability
CS	Cyber Security
DCP	Discovery and Configuration Protocol
DoS	Denial of Service
ETSI	Europäische Institut für Telekommunikationsnormen
FF	Forschungsfrage
H/W	Hardware
HiL	Hardware-in-the-Loop
HTML	Hypertext Markup Language
I-PC	Industrie Personal Computer
I/O	Input/Output
IBN	Inbetriebnahme
IIOT	Industrial Internet of Things
KNN	künstliches neuronales Netz
LED	Light Emitting Diode
MAC	Media Access Control
MiL	Model-in-the-Loop
PLC	Programmable Logic Controller
QS	Qualitäts Sicherung
RDF	Resource Definition Framework
S/W	Software
SAT	Siemens Automation Tool
SCADA	Supervisory Control and Data Acquisition
SiL	Software-in-the-Loop
SPS	Speicherprogrammierbare Steuerung
SVM	Support-Vektor-Maschinen
TA	Teilaufgabe
TIA-Portal	Totally Integrated Automation
VIBN	Virtuelle Inbetriebnahme
XML	Extensible Markup Language

1 Motivation

Moderne Trends in der Fertigung sind durch Massenanpassung, kleine Losgrößen, hohe Variabilität der Produkttypen und ein sich während des Lebenszyklus einer industriellen Anlage änderndes Produktportfolio gekennzeichnet (Luder et al., 2005). Diese Trends implizieren komplexere Anlagen (Mcfarlane und Bussmann, 2000), die Änderungen im physischen Layout einschließlich umfangreicher technischer Aktualisierungen unterstützen. Die Komplexität der Anlagen, einschließlich der Automatisierungshardware und der Automatisierungssoftware, nimmt zu. Da der Anteil der Systemfunktionalität, der durch Software realisiert wird, steigt, sind Konzepte zur Unterstützung der Automatisierungsingenieure im Umgang mit dieser Komplexität dringend erforderlich (Thramboulidis, 2010).

Automatisierte Tests können dazu beitragen, die erforderlichen Ressourcen für die Softwareentwicklung zu minimieren. Änderungen an der Software machen jedoch eine erneute Bewertung der Funktionalität durch Tests erforderlich. Um den Ressourcenverbrauch zu verringern, können bestehende relevante Tests erneut ausgeführt werden, nachdem ihre Kompatibilität mit der Software nach den Änderungen sichergestellt wurde (Ulewicz, Schütz und Vogel-Heuser, 2014). Werden eine Software oder ihre Umgebung geändert, muss zum einen überprüft werden, ob die gewünschte Funktion erfüllt ist, und zum anderen, ob es ungewollte Änderungen oder Seiteneffekte gibt (Zeller, 2019).

Bei der Abarbeitung einer Automatisierungsaufgabe interagieren eine Vielzahl von Steuerungskomponenten verschiedener Hersteller miteinander, wodurch komplexe Abhängigkeiten zwischen den Softwarekomponenten entstehen. Diese Heterogenität führt dazu, dass dem Anlagenbetreiber die Abhängigkeiten innerhalb des Steuerungssystems oftmals unbekannt sind. Erschwerend kommt hinzu, dass sich Abhängigkeiten durch Integration, Entfernung oder Änderung von Komponenten verändern. Dies resultiert darin, dass das Pflegen von Systemmodellen eines Steuerungssystems hohe Expertise und hohen Aufwand erfordert und deshalb meist keine aktuellen Abhängigkeitsmodelle existieren.

Aus diesen Gründen werden aktuell, gemäß dem allgemeinen Instandhaltungsgrundsatz, „never touch a running system“, Änderungen an industri-

ellen Anlagen gemieden (Zeller und Weyrich, 2016) . Wie in Abbildung 1.1 dargestellt, spielt deshalb das Testen während der Betriebsphase eine untergeordnete Rolle. Während des Betriebs werden neben der Produktion hauptsächlich qualitätssichernde Maßnahmen (QS-Maßnahmen) betrieben. Das Testen ist heute hauptsächlich Bestandteil des Engineerings und der (Wieder-) Inbetriebnahme (IBN) (Zeller, 2019).

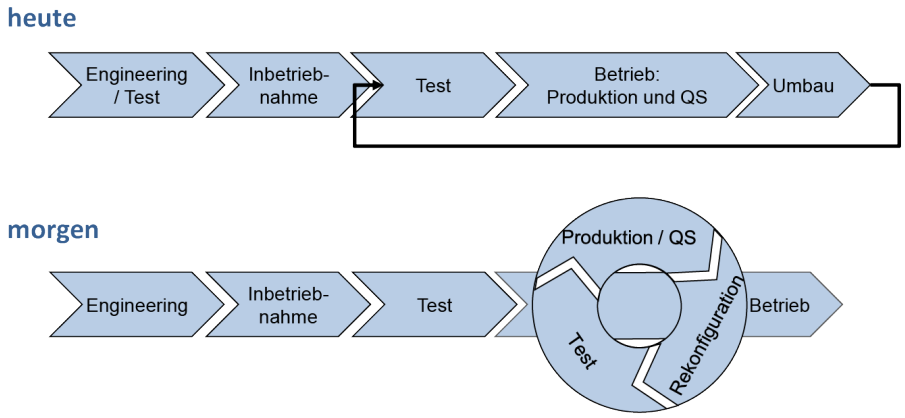


Abbildung 1.1: Änderung des Lebenszyklus mit VIBN nach (Zeller, 2019)

In Zeiten der voranschreitenden Digitalisierung und Vernetzung von industriellen Systemen werden die Austauschbarkeit, Interoperabilität und Robustheit gegen Fehler bei Hard- und Softwarekomponenten zunehmend zu einem wichtigen Thema (acatech, 2013). Speziell die eingesetzten Steuerungssysteme müssen dementsprechend gegen Änderungen der Hardware und Software abgesichert werden. Dabei nehmen Konzepte wie Virtuelle Inbetriebnahme (VIBN) und Cyber Security (CS) Schlüsselrollen bei der Entwicklung und Absicherung von Hard- und Softwarekomponenten ein. Flexible Kundenwünsche, sich ständig ändernde Marktanforderungen und der weltweite Wettbewerb erfordern hochflexible industrielle Anlagen (Koren et al., 1999). Um die Kompatibilität der verbauten Hard- und Softwarekomponenten speziell beim Aufspielen neuer Software bzw. beim Neustart der Hardwarekomponenenten sicherzustellen, müssen Teststrategien und Konzepte entwickelt werden, welche dem Anlagenbetreiber eine möglichst einfache Fehlererkennung und Beseitigung ermöglichen (Zeller, 2019).

1.1 Aufgabenanalyse

In der Aufgabenanalyse wurden Teilaufgaben (TA) aus den wissenschaftlichen Fragestellungen (FF) der Aufgabenstellung extrahiert. Diese Teilaufgaben ebnen den Weg für die Gliederung dieser Arbeit und werden im nachfolgenden kurz beleuchtet. Weiterhin wird erläutert in welchen Aufgabenkomplexen die aufgestellten TA beantwortet werden.

1.1.1 Forschungsfragen und Teilaufgaben

FF 1: Welche wesentlichen Teststrategien, beispielsweise aus der VIBN oder der CS gibt es, und wie lassen sie sich auf Software-Upload bzw. Software-Neustart übertragen, um die Kompatibilität von Software und Hardware sicherzustellen?

TA 1 Als erste TA ergibt sich eine durch Literaturrecherche gestützte Definition der Begriffe „VIBN“ und „CS“.

TA 2 Als zweite TA steht die Bestimmung bestehender automatisierbarer Teststrategien aus VIBN und CS, die eine Kompatibilität von Software und Hardware sicherstellen.

TA 3 Die dritte TA umfasst die Bestimmung, wie Teststrategien auf den Software-Upload bzw. Neustart übertragen werden können.

TA 4 Als vierte TA ergibt sich die Feststellung der Eignung gefundener Teststrategien.

FF 2: Welche wesentlichen Eigenschaften bzw. Informationen müssen Software- und Hardwarekomponenten aufweisen, um einen Kompatibilitätstest zu ermöglichen?

TA 5 Die fünfte TA besteht aus der Bestimmung, was „wesentliche“ Eigenschaften und Informationen sind und wie diese ermittelt werden können.

TA 6 Als sechste TA steht die auf der Literaturrecherche basierende Feststellung aller Eigenschaften und Informationen der benötigten Software- und Hardwarekomponenten.

TA 7 Als siebte TA soll bestimmt werden, welche Eigenschaften und Informationen für einen Kompatibilitätstest der Software mit der Hardware nötig sind.

FF 3: Wie kann ein Konzept aussehen, welches angemessene Reaktionen auf gefundene Inkompatibilitäten bzw. Fehler ermöglicht?

TA 8 Hier muss zunächst bestimmt werden, wie Inkompatibilitäten bzw. Fehler im Kontext der Arbeit definiert sind.

TA 9 Die neunte TA umfasst eine Bestimmung möglicher Reaktionen auf Inkompatibilitäten bzw. Fehler.

TA 10 Als zehnte TA ergibt sich die Erstellung eines Konzeptes zur Kompatibilitätsprüfung, sowie ein Konzept, um auf gefundene Inkompatibilitäten zu reagieren.

TA 11 Als elfte TA folgt die Bestimmung einer Möglichkeit auf gefundene Inkompatibilitäten zu reagieren.

TA 12 Die zwölfte und letzte TA beschäftigt sich mit der abschließenden kritischen Bewertung der Arbeitsweise und der Forschungsergebnisse.

1.1.2 Aufgabenkomplexe

Aus den einzelnen Teilaufgaben und Forschungsfragen wurden die folgenden Aufgabenkomplexe abgeleitet. Die Aufgabenkomplexe bilden die Struktur dieser Arbeit und beantworten die im vorangegangen Kapitel aufgestellten Teilaufgaben.

Literaturrecherche In diesem ersten Aufgabenkomplex wird zunächst eine systematische Literaturrecherche mittels vorher ausgewählter Suchbegriffe durchgeführt. Die dadurch gefundene Literatur wird dann durch das Schneeballsystem weitergehend ergänzt. Im Verlauf der Arbeit wird auf die in diesem Komplex gefundene Literatur immer wieder zurückgegriffen, sodass dieser Komplex auch eine strukturierte Verwaltung der Literatur mittels Software erfordert. Durch die Literaturrecherche soll eine gute Basis an fundierten Quellen aufgebaut werden, um diese dann zu analysieren und anzuwenden.

Analyse der Literatur und Stand der Technik Im an die Literaturrecherche anschließenden Komplex erfolgt eine Analyse der gefundenen Literatur und eine tiefergehende Auseinandersetzung mit den gestellten wissenschaftlichen Forschungsfragen und den Teilaufgaben. Das Ergebnis dieses Aufgabenkomplexes ist die Darlegung des aktuellen Stand der Technik (siehe Kap. 2). Darin werden die TA 1 und TA 2 beantwortet.

Anforderungsanalyse Im Aufgabenkomplex der Anforderungsanalyse (siehe Kap. 3) werden, aufbauend auf dem Stand der Technik, die Anforderungen an das Konzept erstellt und die TA 3-7 beantwortet.

Erstellung des Teststrategienkonzeptes Nach gründlicher Analyse der Literatur und Auseinandersetzung mit dem gestellten Thema, werden die TA 8-11 bearbeitet. Dabei wird das Konzept (siehe Kap. 4) einer Teststrategie für die Kompatibilitätsprüfung von Hardware und Software erstellt.

Abschließende Bewertung In der abschließenden kritischen Auseinandersetzung, wird durch eine Konzeptevaluation (siehe Kap. 5) und der Bewertung der gewählten Arbeitsweise (siehe Kap. 6) die Arbeit abgeschlossen. Die TA 12 wird in diesem letzten Aufgabenkomplex bearbeitet.

1.1.3 Eingrenzung der Aufgabenstellung

Das Thema der Kompatibilitätstests für Software- und Hardwarekomponenten in industriellen Steuerungen ist sehr breit gefächert und muss zunächst eingegrenzt werden. Ein Aspekt, der in dieser Arbeit besonders betrachtet werden soll, ist die Erarbeitung einer Teststrategie zur Prüfung der Kompatibilität von Hardware und Softwarekomponenten in einer einzelnen industriellen Steuerung und deren Umgebung. Weiterhin sollen die Inkompatibilitäten und Fehler, die auftreten können, genannt werden und eine Auflistung möglicher Reaktionen auf diese erarbeitet werden. Durch übersichtliche Tabellen zur Darstellung der Fehler soll das Ergebnis der Kompatiblitätsprüfung abgerundet werden.

Um die Kompatibilität zweier Komponenten testen zu können, müssen die Soll- und Ist-Werte dieser vorliegen. In der Arbeit sollen die Wege zur Bestimmung dieser Soll- und Ist-Werte ebenfalls aufgezeigt werden und anhand eines industriellen Referenzsystems erläutert werden.

Informationen über ein passendes Referenzsystem konnten durch den Austausch mit Mitarbeitern der Firma Sartorius[1] ermittelt werden.

1.1.4 Ziel der Arbeit

Ziel dieser Studienarbeit ist die Erstellung einer Teststrategie, welche sowohl beim Neustart von industriellen Steuerungen, als auch beim Aufspielen neuer Software automatisiert ablaufen kann und die Inkompatibilität der einzelnen Hard- und Softwarekomponenten aufzeigt.

1.1.5 Festsetzung von Formulierungen

Um einheitliche Formulierungen in dieser Arbeit zu gewährleisten, werden folgende Festlegungen gemacht.

„Speicherprogrammierbare Steuerungen“ (SPS) sind per Definition industrielle Steuerungen und werden daher in dieser Arbeit synonym verwendet. Von nun an wird nur ausschließlich die Abkürzung „SPS“ genutzt, um auf eine industrielle Steuerung zu verweisen.

Da die zu ermittelnde Teststrategie in dieser Arbeit die Prüfung auf Inkompatibilitäten umfasst, werden die Wörter „Teststrategie“ und „Kompatibilitätsprüfung“ gleichbedeutend angesehen. Zur Vereinheitlichung wird daher nur noch das Wort „Kompatibilitätsprüfung“ genutzt, um auf die Teststrategie zu verweisen.

[1] Web: https://sartorius.com

2 Einführung und Stand der Technik

Dieses Kapitel führt in den aktuellen Stand der Technik im Bereich der VIBN, Soft- und Hardwaretests und Themen der CS ein. Weiterhin wird der Begriff der Kompatibilität basierend auf Literaturquellen definiert. Zum Ende der Arbeit wird noch einmal auf den Stand der Forschung eingegangen und weiterer Forschungsbedarf aufgezeigt.

2.1 VIBN von industriellen Anlagen

Die VIBN ist bereits seit vier Jahrzehnten Gegenstand von Studien (Hoffmann et al., 2010). Vereinfacht ausgedrückt handelt es sich dabei um den Einsatz von Simulationstechniken, um das Systemverhalten mit einem virtuellen Maschinenmodell zu testen, bevor es an das reale System angeschlossen wird (Ugarte et al., 2022).

Viele Unternehmen entwickeln ihre Maschinen und Werkzeuge immer noch unsynchronisiert, die Ergebnisse der Konstruktions- und Engineering-Phasen kollidieren daher oft bei der ersten IBN (Lechler et al., 2019). Mithilfe der VIBN unter Verwendung eines Simulationsmodells kann die Industrie diese Kollision vermeiden. Das Konzept der VIBN wird aber bisher in der Industrie nur wenig umgesetzt. Die VIBN hilft die Projekte im Zeitplan zu halten, verbessert die Effizienz beim Engineering und reduziert Kosten und Zeit während der realen IBN (C. G. Lee und Park, 2014).

Eine experimentelle Studie zur VIBN (Zäh et al., 2006) zeigt die positiven Auswirkungen der VIBN auf die Fehlerquote bei der realen IBN. Die Studie wurde mit zwei Gruppen von Kontrollprogrammierern durchgeführt. Jede Gruppe bestand aus 30 Personen. Eine Gruppe wandte VIBN bei der Softwareentwicklung für eine Maschine an. Die Ergebnisse wurden mit denen der zweiten Gruppe von Programmierern verglichen, die keine VIBN anwandten. Als Prüfstand diente eine Blechdosenpresse mit einer Siemens S7-300 SPS. Eine Gruppe programmierte die SPS und testete das Programm anschließend in einer realen IBN an einer realen Maschine. Die andere Gruppe programmierte mit Hilfe eines virtuellen Maschinenmodells entsprechend der VIBN. Sie führten die reale IBN nicht durch, bevor sie eine erfolgreiche VIBN erreicht hatten. Die Ergebnisse zeigten eine Verkürzung der realen

Inbetriebnahmezeit um 75 %, was auf eine verbesserte Softwarequalität zu Beginn der realen IBN zurückzuführen ist. Dies unterstreicht die Vorteile der Durchführung einer VIBN, aber das Modell der virtuellen Maschine war bereits im Vorfeld dieser Studie entwickelt worden. Dieser Aufwand wurde nicht in Betracht gezogen.

In der Phase der Softwareentwicklung und der Vorinbetriebnahme ist das manuelle Überprüfen der Hardwarekomponenten sehr schwierig oder sogar unmöglich. Trotz der Bemühungen, die Validierung durchzuführen, besteht aufgrund des hohen Arbeitsaufwands die Gefahr, dass Fehler übersehen werden. Infolgedessen können Designprobleme und Fehler vor der ersten IBN unentdeckt bleiben, was zeit- und kostenintensive Korrekturmaßnahmen nach sich zieht (Bartz et al., 2021).

Der Trend der VIBN geht dahin, dass diese von Anfang an Teil der Entwicklung neuer Werkzeuge und Systeme sein wird. Mit Hilfe von automatisierten Testlösungen wird es möglich sein, die Zuverlässigkeit des Systems während seiner gesamten Lebensdauer zu erhalten und zu erweitern. Die Herausforderung besteht darin, die anfänglichen Einführungskosten zu senken und die Systeme zunehmend modular und generisch zu gestalten (Bartz et al., 2021).

Da die VIBN parallel zum Produktions- und Montageprozess durchgeführt werden kann, lässt sich durch Optimierungsschleifen während der in Abb. 2.1 durch Pfeile gekennzeichneten Prozesse wertvolle Zeit im anstehenden Inbetriebnahmeprozess einsparen (Lechler et al., 2019).

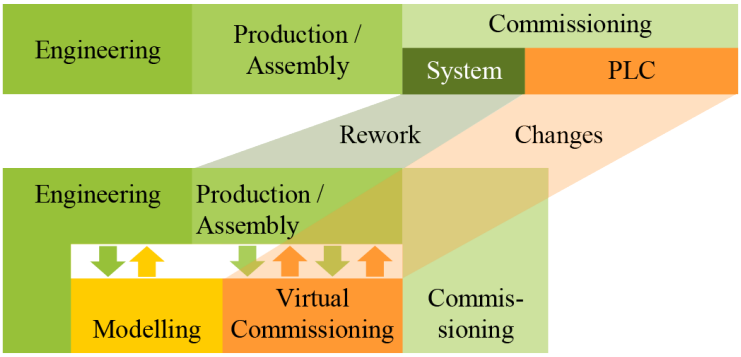


Abbildung 2.1: Engineering mit und ohne VIBN (Lechler et al., 2019)

2.1.1 Teststrategien aus der VIBN

In diesem Kapitel werden drei Teststrategien aus dem Bereich der VIBN näher erklärt.

2.1.1.1 Model-in-the-Loop

Beim sogenannten Model-in-the-Loop (MiL)-Ansatz werden alle Komponenten einer Anlage modelliert und die Simulation wird innerhalb der modellbasierten Simulationsumgebung durchgeführt. Diese Art der Simulation wird vor allem in frühen Phasen des Projekts eingesetzt (Hill et al., 2021).

Dieser Ansatz hat, je nach Anwendung, eine Reihe potenzieller Vorteile (Plummer, 2006):

1. Nur die Schlüsselkomponente mit unbekannter Dynamik muss physisch getestet werden, wodurch die Kosten und Komplexität der physischen Prüfeinrichtung sinken.

2. Ein neu entworfenes System kann getestet werden, auch wenn einige Teile noch nicht physikalisch realisiert sind. Diese Teile werden stattdessen simuliert.

3. Manchmal ist es schwierig, die Betriebsbedingungen im Labor nachzubilden, z. B. Umgebungsbedingungen oder aerodynamische Kräfte, so dass die betroffenen Systeme durch ein Computermodell genauer dargestellt werden.

4. Die Eigenschaften des simulierten Systems können variiert werden, um alternative Konfigurationen darzustellen. Dies ist effizienter als die Änderung physischer Komponenten.

Abbildung 2.2 zeigt die Simulationskonzepte der VIBN.

Bei der MiL-Simulation werden sowohl die Anlage als auch die Steuerung in Simulationsmodelle umgewandelt, verknüpft und dann in einem geschlossenen Kreislauf simuliert (Machado und Seabra, 2013).

2.1.1.2 Software-in-the-Loop

Beim Software-in-the-Loop (SiL) Ansatz wird eine virtuelle SPS instanziiert, um den Automatisierungscode zu testen, der mit den Verhaltensmodellen in der Simulationsschicht verbunden ist (Hill et al., 2021).

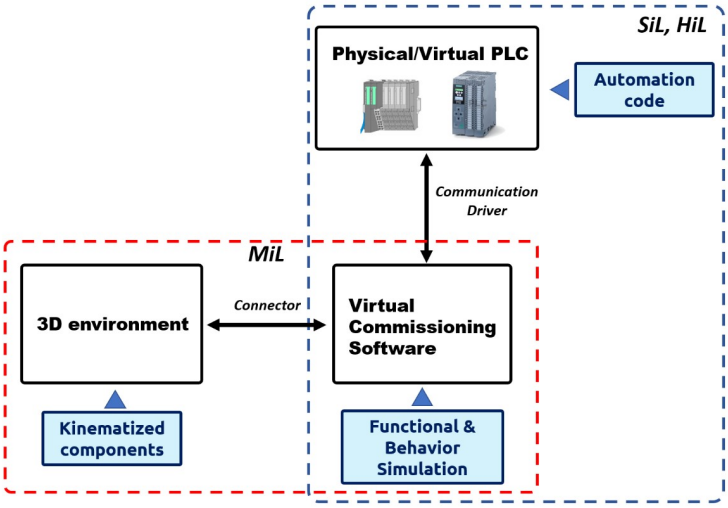


Abbildung 2.2: MiL, SiL und HiL Simulationsübersicht von (Hill et al., 2021)

Dieser Ansatz erlaubt es, Software-Komponente mit einer Umgebungssimulation zu integrieren (Muresan und Pitica, 2012). Außerdem ermöglicht dieser Ansatz das sehr schnelle Testen verschiedener Szenarien und Steuerungsalgorithmen und deren Flexibilität.

Die Kosten für die Implementierung einer SiL-Umgebung sind etwa sechzigmal geringer als bei einer Hardware-in-the-Loop (HiL)-Umgebung. Die SiL Umgebung kann bei jedem Entwickler vorhanden sein, während für HiL eine separate Ausrüstung benötigt wird (Muresan und Pitica, 2012).

SiL-Tests werden durchgeführt, indem die Software auf einer normalen PC-Hardware ausgeführt wird, die es ermöglicht, die wichtigsten Fehler im Funktionsbereich zu identifizieren. Allerdings können sich der Compiler und der Prozessor eines PCs anders verhalten als auf der Plattform (Mazza, 2018).

2.1.1.3 Hardware-in-the-Loop

Beim Hardware-in-the-Loop (HiL) Ansatz wird eine reale physikalische SPS mit einer Simulationsschicht verbunden, welche die Verhaltensmodelle der Anlage ausführt.

Alle Prozesse der VIBN basieren auf einem virtuellen Modell, das mit einer SPS verbunden ist. Im Falle einer Hardware-in-the-Loop (HiL)-Simulation ist die SPS eine reale Hardware-Steuerung (Oppelt, Wolf und Urbas, 2015). Folglich ist es möglich, die VIBN mit der SPS durchzuführen, die anschlie-

ßend in das Produktionssystem integriert wird. Nach Mazza (Mazza, 2018) ist dies besonders interessant für:

1. Validierung von SPS-Steuerungsstrategien auf der Grundlage einer virtualisierten Umgebung mit der Fähigkeit, die erwartete Dynamik der realen Maschine darzustellen.

2. Verbesserung oder Vergleich von in der Praxis gemessenen Daten mit simulierten Daten (z.B. von virtuellen Sensoren).

3. Unterstützung der Bediener während des realen Maschinenbetriebs durch simulierte Vorhersagen oder Diagnosen, die von einem „digitalen Zwilling “ mit realen Daten aus dem Feld gespeist werden.

Zusammenfassend lassen sich folgende Gründe finden, warum die VIBN Strategien SiL und HiL sehr nützlich sind:

1. Steuerungsstrategien können virtuell validiert werden, ohne Menschenleben oder Maschinen zu gefährden.

2. Die Kosten können dank der Möglichkeit der Fehlersuche gesenkt werden (es könnte die Fehlerkorrektur während des Entwurfsprozesses zu spät erfolgen).

3. Bediener können sich mit den Steuerungssystemen vertraut machen, auch mit den im Bau befindlichen, dank der Schaffung von virtuellen Systemen.

4. Fehler können innerhalb weniger Minuten mit Hilfe der „virtuellen Zeit“ durch Simulation gefunden werden.

2.1.1.4 Konklusion und Forschungsbestrebungen

In Abb. 2.3 sind die grundlegenden Unterschiede von den Konzepten der VIBN zusammengestellt. Alle Konzepte der VIBN basieren auf einem virtuellen Modell, das mit einer SPS verbunden ist. Im Falle einer HiL-Simulation ist die SPS eine reale Hardware-Steuerung (Oppelt, Wolf und Urbas, 2015). Folglich ist es möglich, die VIBN mit der SPS durchzuführen, die anschließend in das Produktionssystem integriert wird. Andererseits kann die VIBN auch auf eine emulierte SPS angewendet werden, was als SiL-Simulation bezeichnet wird. Daher ist keine Hardware-SPS erforderlich, was einer der

Hauptvorteile von SiL-Simulationen ist. Außerdem ist es möglich, eine Reality-in-the-Loop-Simulation durchzuführen, bei der eine emulierte SPS mit dem Produktionssystem kombiniert wird, um bestimmte reale Komponenten zu testen (C. G. Lee und Park, 2014). Das virtuelle Modell kann mit Komponenten der realen Maschine kombiniert werden, um deren zukünftige Funktion zu testen, was zu einer hybriden Simulation führt. Darüber hinaus ist es möglich die VIBN in verschiedenen Software-Tools durchzuführen. Sie alle haben ihre Vorteile in Bezug auf Leistung, Zuverlässigkeit und Benutzerfreundlichkeit. Ein genauer Vergleich der verfügbaren Tools ist ein zukünftiges Feld der Forschung und sollte genauer untersucht werden (Lechler et al., 2019).

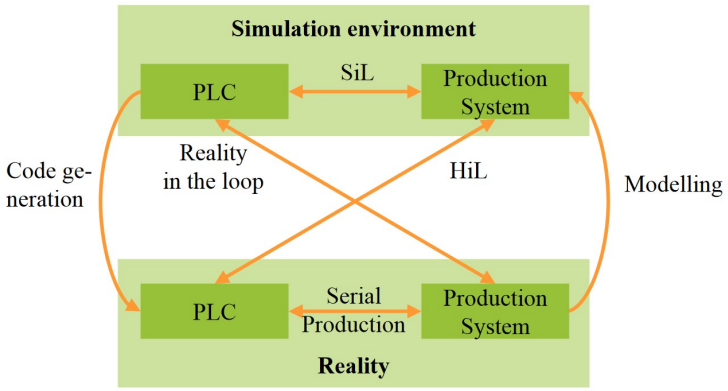


Abbildung 2.3: Teststrategien der VIBN (Lechler et al., 2019)

2.2 CS in industriellen Anlagen

Während IIoT, Cloud Computing und künstliche Intelligenz die Innovationen im Fertigungssektor vorantreiben, sind die Hersteller zunehmend anfällig für Cyberangriffe (Thames und Schaefer, 2017). Einem Bericht von Accenture und dem Ponemon Institute (Maurushat, 2013) zufolge, liegen die durchschnittlichen Kosten für Cyber-Kriminalität weltweit auf 11,7 Millionen US-Dollar pro Unternehmen im Jahr 2017. Cyber-Bedrohungen reichen von Angriffen auf Computer, Netzwerke, Smartphones und Stromnetze bis hin zur industriellen Fertigung. Nach Angaben von NBC News verlor das verarbeitende Gewerbe von 2002 bis 2012 in den USA fast 240 Milliarden Dollar an Einnahmen und 42.220 Arbeitsplätze aufgrund von Cyberangriffen (NBC News, 2014). Einer der Hauptgründe, warum der Fertigungssek-

tor nach dem Gesundheitswesen zu den am häufigsten gehackten Branchen gehört, liegt vor allem an IIoT-verbundene Maschinen, Cloud-basierte Fernerkundung und intelligenten Steuerungssystemen. So wurde beispielsweise Stuxnet, ein bösartiger Computerwurm, der erstmals 2010 entdeckt wurde, entwickelt, um SCADA Systeme und SPS anzugreifen (Langner, 2011). Stuxnet zerstörte fast ein Fünftel der iranischen Atomzentrifugen, indem er über 200.000 Computer infizierte und 1.000 Maschinen physisch lahmlegte. Im Jahr 2014 hackten Angreifer das Steuerungssystem eines deutschen Stahlwerks mit Hilfe von gefälschten E-Mails. Aus einem Bericht des Bundesamtes für Sicherheit in der Informationstechnik (BSI, 2014) geht hervor, dass das Kontrollsystem des attackierten Werks aufgrund dieses Cyberangriffs nicht in der Lage war, einen Hochofen ordnungsgemäß abzuschalten.

Nach Angaben des U.S. National Center for Manufacturing Science machten Varianten von Trojanern und Droppern 86 % der Malware im Fertigungssektor aus (B. Raymond, 2016).

2.2.1 Sicherheitsziel

Das wichtigste Sicherheitsziel ist der Schutz der Vertraulichkeit, Integrität und Verfügbarkeit (auch bekannt als CIA-Dreiklang) von Daten. Vertraulichkeit bedeutet, dass sensible Daten und Informationen nicht an Unbefugte weitergegeben werden dürfen. Integrität bedeutet, die Konsistenz, Genauigkeit und Vertrauenswürdigkeit von Daten zu erhalten. Bei der Verfügbarkeit geht es darum, Daten und Ressourcen für die autorisierte Nutzung verfügbar zu halten (Warsinske, 2019).

2.2.2 Teststrategien aus der CS

Die Erkennung von Eindringlingen ist ein Prozess, bei dem die Aktivitäten in einem Netz oder Computersystem auf mögliche Sicherheitsprobleme überwacht werden. Die Erkennung von Eindringlingen umfasst die Überwachung von Systemaktivitäten, die Prüfung von Systemschwachstellen, die statistische Analyse von Aktivitätsmustern und die Analyse abnormaler Aktivitäten. Die Methoden zur Erkennung von Eindringlingen lassen sich in zwei Kategorien einteilen: signaturbasierte und anomaliebasierte Erkennung (Kaur, M. Kumar und Bhandari, 2017).

2.2.2.1 Signaturbasierte Erkennung

Bei der signaturbasierten Erkennung, die auch als Missbrauchserkennung bezeichnet wird, werden bekannte Angriffe anhand des Systemverhaltens erkannt (Wu et al., 2018). Zu den signaturbasierten Erkennungsmethoden gehören die Zustandsübergangsanalyse und Petri-Netze. Ilgun et al. (Ilgun, R. Kemmerer und Porras, 1995) stellten einen Ansatz zur Darstellung und Erkennung von Computereinbrüchen mithilfe eines Zustandsübergangsdiagramms vor. Dieser Ansatz modelliert Eindringlinge als eine Reihe von Zustandsänderungen, die von einem sicheren Ausgangszustand zu einem kompromittierten Zielzustand führen. Vigna und Kemmerer (Vigna und R. Kemmerer, 1998) entwickelten einen netzwerkbasierten Ansatz zur Erkennung von Eindringlingen, der auf einer erweiterten Technik zur Analyse von Zustandsübergängen (d. h. NetSTAT) basiert. NetSTAT wurde verwendet, um ein formales Modell des Angriffsszenarios mithilfe eines Zustandsübergangsdiagramms zu erstellen. Anhand des formalen Modells des Angriffs ist NetSTAT in der Lage zu bestimmen, welche Netzwerkereignisse überwacht werden sollten. Ho et al. (Ho, Frincke und Tobin, 1998) schlugen einen Ansatz zur Erkennung von Eindringlingen vor, der partielle Ordnungsplanung und ausführbare Petri-Netze kombiniert. Dieser Ansatz ist in der Lage, Angriffe mit einem vorher festgelegten unerwünschten Verhalten oder einer Zustandsänderung zu erkennen. Kumar und Spafford (S. Kumar und Spafford, 1994) entwickelten eine Petri-Netz-basierte Methode zur Intrusion Detection. Das Wissen über Angriffe wurde als farbige Petri-Netze dargestellt. Die Petri-Netze stellen den Übergang von Systemzuständen entlang von Pfaden dar, die zu angegriffenen Zuständen geführt haben.

2.2.2.2 Anomaliebasierte Erkennung

Bei der anomaliebasierten Erkennung werden unbekannte Angriffe mit Hilfe statistischer Methoden und künstlicher Intelligenz aufgedeckt (Wu et al., 2018). Ourston et al. (Ourston et al., 2003) stellten einen Ansatz vor, der versteckte Markov-Modelle zur Erkennung komplexer Internet-Angriffe verwendet. Diese Methode ist in der Lage, das Problem der mehrstufigen Angriffe anzugehen. Experimentelle Ergebnisse haben gezeigt, dass diese Methode effektiver ist als klassische maschinelle Lerntechniken, wie Entscheidungsbäume und künstliche neuronale Netze (KNN). Mukkamala et al. (Mukkamala, Janoski und Sung, 2002) entwickelten eine Methode zur Erkennung von Angriffen unter Verwendung von KNNs und Support-Vektor-Maschinen (SVMs).

KNNs und SVMs wurden verwendet, um die Klassifikatoren anhand einer Liste von Merkmalen zu erstellen. Experimentelle Ergebnisse haben gezeigt, dass KNNs und SVMs in der Lage sind, Anomalien und bekannte Eindringlinge zu erkennen. Pan et al. (Pan et al., 2003) entwickelten eine hybride Methode zur Erkennung von Angriffen durch die Kombination von KNNs und Entscheidungsbaum-Algorithmen. Experimentelle Ergebnisse haben gezeigt, dass KNNs DoS- und Sondierungsangriffe effektiver erkennen können, als die Erkennung von unautorisiertem Zugriff von einem entfernten Rechner und autorisiertem Zugriff auf lokale Superuser-Angriffe. Zhang et al. (J. Zhang et al., 2008) entwickelten eine auf Zufallswäldern basierende Methode zur Erkennung von Netzwerkeinbrüchen. Diese Methode wurde an einem Datensatz zur Erkennung von Einbrüchen demonstriert. Die experimentellen Ergebnisse haben gezeigt, dass die vorgeschlagene Methode eine hohe Erkennungsrate mit einer niedrigen Falsch-Positiv-Rate erreichen kann. Gaddam et al. (Gaddam, Phoha und Balagani, 2007) entwickelten einen Ansatz zur Erkennung von Anomalien unter Verwendung von kaskadierenden K-Means-Clustering- und ID3-Entscheidungsbaum-Lernalgorithmen. Diese Methode wurde zur Analyse eines Datensatzes von Netzwerkanomalien verwendet. Experimentelle Ergebnisse haben gezeigt, dass die Erkennungsgenauigkeit bis zu 96,24 % bei einer Falsch-Positiv-Rate von 3 % beträgt. Liao und Vemuri (Liao und Vemuri, 2002) entwickelten einen Klassifikator zur Erkennung von Eindringlingen unter Verwendung des k-Nächste-Nachbarn-Algorithmus (kNN). Diese Methode wurde verwendet, um das Verhalten von Programmen als normal oder störend zu klassifizieren. Experimentelle Ergebnisse haben gezeigt, dass der kNN-Klassifikator Angriffe mit einer niedrigen Falsch-Positiv-Rate effektiv erkennen kann. Sabhnani und Serpen (Sabhnani und Serpen, 2003) analysierten einen Datensatz zur Erkennung von Eindringlingen mit Hilfe einer Reihe von Algorithmen für maschinelles Lernen. Der Datensatz umfasst vier Arten von Hauptangriffen, darunter Sondierungs-, DoS-, User-to-Root- und Remote-to-Local-Angriffe. Simulationsergebnisse haben gezeigt, dass bestimmte Klassifizierungsalgorithmen für eine bestimmte Angriffskategorie effektiver sind. Lee et al. (K. Lee et al., 2008) führten eine auf Clusteranalyse basierende Angriffserkennungsmethode ein, um DoS-Angriffe proaktiv zu erkennen. Ein hierarchischer Clustering-Algorithmus wurde verwendet, um einen Datensatz zur Erkennung von Angriffen zu analysieren. Experimentelle Ergebnisse haben gezeigt, dass diese Methode in der Lage ist, DoS-Angriffe zu erkennen.

2.2.2.3 Konklusion und Forschungsbestrebungen

Die signaturbasierte und anomaliebasierte Erkennung eignet sich gut, um sicherheitsrelevante Probleme aufzudecken. Der Einsatz bei der Erkennung von Kompatibilität von Software und Hardware ist aber nicht gegeben.

Zukünftige Forschungen können die Studie von Lezzi et al. (Lezzi, Lazoi und Corallo, 2018) als Referenzrahmen nutzen, um Untersuchungen im industriellen Bereich durchzuführen und den aktuellen Stand der Technik zu erweitern. Aus Standpunkt des Managements gesehen, bietet die Studie eine Abkürzung zu einem vollständigen Überblick über die Cybersicherheit in der I-4.0. Diese kann als Grundlage für die Unterstützung des Entscheidungsfindungsprozesses über Cybersicherheitsfragen, aber auch als Referenzmaterial für Ausbildungen in der IT-Abteilung dienen.

2.3 Interoperabilität als Kompatibilitätsmaß

Um Kompatibilitätstests zwischen verschiedenen Komponenten in industriellen Steuerungen durchführen zu können, muss zunächst ein Kompatibilitätsmaß hergeleitet werden. Der Begriff der Interoperabilität wird dabei in der Literatur häufiger verwendet und kann daher im Kontext dieser Arbeit als Synonym für Kompatibilität gesehen werden.

Das Problem der Interoperabilität von Informationssystemen besteht seit 1988 (Helsinki, Eliassen und Veijalainen, o.D.), möglicherweise sogar schon früher. In der Literatur gibt es mehrere Definitionen für Interoperabilität. Aus den verschiedenen Definitionen für Interoperabilität zitieren wir diejenigen, die für unseren Kontext relevant sind. Das Oxford Dictionary definiert Interoperabilität allgemein als „in der Lage sein, zusammen zu arbeiten“. Dies bedeutet, dass zwei interoperable Systeme sich gegenseitig verstehen und die Funktionalität des jeweils anderen nutzen können. ISO/IEC definiert Interoperabilität als „die Fähigkeit, zwischen verschiedenen Funktionseinheiten so zu kommunizieren, Programme auszuführen oder Daten zu übertragen, dass der Benutzer nur wenig oder gar keine Kenntnisse über die besonderen Merkmale dieser Einheiten haben muss“ (ISO/IEC 2382:2015, 2015). Im weiteren Sinne wird Interoperabilität von der IEEE definiert als „die Fähigkeit von zwei oder mehr Systemen oder Komponenten, Informationen auszutauschen und die ausgetauschten Informationen zu nutzen“ (Radatz, Geraci und Katki, 1990). Nach dieser Definition wird die Interoperabilität durch die Entwicklung von Standards realisiert. Im IoT kann Interoperabilität als die Fähigkeit zweier Systeme definiert werden, miteinander zu kommunizieren

und Dienste gemeinsam zu nutzen (Kiljander et al., 2014).

Die Fähigkeit zweier Systeme zur Interoperabilität kann auch mit Hilfe verschiedener Ebenenmodelle dargestellt werden. So wurde beispielsweise von Tolk et al. (Tolk, 2004) eine sechsstufige Struktur ausgearbeitet, die Folgendes umfasst: no connection (keine Interoperabilität zwischen Systemen), technical (grundlegende Konnektivität und Netzkonnektivität), syntactical (Interoperabilität beim Datenaustausch), semantic (Verständnis der Bedeutung der Daten), pragmatic/dynamic (Anwendbarkeit der Informationen) und conceptual (gemeinsame Weltsicht). Ein ähnliches Modell mit sechs Ebenen wird in (Pantsar-Syväniemi et al., 2012) von Pantsar Syvaniemi et al. vorgeschlagen: connection, communication, semantic, dynamic, behavioural, und conceptual. Diese sechs Ebenen entsprechen den technical, syntactical, semantic, pragmatic/dynamic bzw. conceptual Ebenen des Modells von Tolk.

2.4 Testautomatisierung und Test Case Generierung

Es besteht weitgehend Einigkeit darüber, dass eine vollständige Automatisierung die Testkosten senkt, insbesondere wenn es um Regressionstests geht. So stimmten beispielsweise 64 % der Befragten in einer Literaturübersicht und einer Umfrage unter Praktikern dieser Aussage zu (Rafi et al., 2012). Insbesondere gibt es einige empirische Belege aus einem Fallstudienbericht im Kontext der agilen Entwicklung (Collins und Lucena, 2012).

2.5 Allgemeine Softwareteststrategien

2.5.1 Modellbasiertes Testen

Die beiden bekanntesten Methoden der formalen Verifikation sind das Beweisen von Theoremen und die Modellprüfung. Diese Methoden werden in einem breiten Kontext in der SPS-Softwareverifikation angewandt. Die Modellprüfung ist eine weit verbreitete formale Methode, bei der das zu verifizierende System durch ein geeignetes Modell dargestellt wird und die gewünschte zu verifizierende Eigenschaft durch systematische Erkundung aller möglichen Zustände, die das modellierte System durchlaufen kann, in einer Brute-Force-Methode überprüft wird. Durch die Berücksichtigung aller möglichen Szenarien kann die verifizierte Eigenschaft in Abhängigkeit von der Korrektheit des Systemmodells garantiert werden (Ovatman et al., 2014).

Bei der Verwendung dieser modellbasierten Prüfung zur Testfallgenerierung reichen die Testaktivitäten von der Beschreibung testrelevanter Anforderungen bis hin zur Testdurchführung. Das gesamte Vorgehen besteht aus den Schritten Modellierung, Testfallgenerierung, Testdurchführung und Auswertung (siehe Abb. 2.4) (Magnus, Russ und Krause, 2016).

Modellbasierte Verfahren versprechen eine höhere Testabdeckung durch die automatische Generierung der Testfälle. Dafür wird allerdings ein Modell der Anforderungen, das Spezifikationsmodell, benötigt. Es beschreibt das Verhalten des Testobjekts aus Sicht des Testers. Die Erstellung der Spezifikationsmodelle wird in der Praxis allgemein als sehr schwierig wahrgenommen (Magnus, Russ und Krause, 2016).

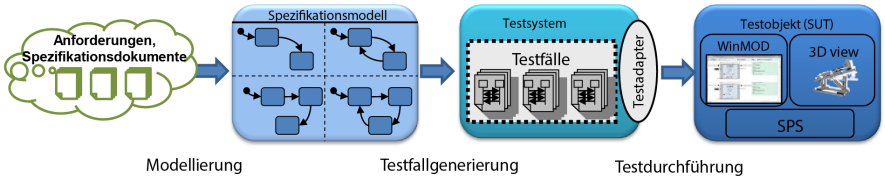


Abbildung 2.4: Testaktivitäten im modellbasierten Testprozess (Magnus, Russ und Krause, 2016)

2.5.2 Funktionale Tests

Heutzutage ist der Funktionstest ein wesentlicher Bestandteil des Entwicklungsprozesses einer industriellen Anlage. In der Regel folgt dieser Entwicklungsprozess dem V-Modell (siehe Abb. 2.5), bei dem Unittests, Integrationstest, Systemtest und Abnahmetest definiert sind. Der Testprozess beginnt mit dem Unit-Testing. Dies geschieht meist mit White-Box-Testing-Techniken durch Evaluierung des Codes. Integrationstests validieren die korrekte Interaktion zwischen verschiedenen Einheiten. Der Systemtest verifiziert das gesamte System und wird normalerweise mit Black-Box-Testing-Techniken durchgeführt und sollte von Ingenieuren durchgeführt werden, die nicht am Entwicklungsprozess beteiligt waren (Zeller und Weyrich, 2016).

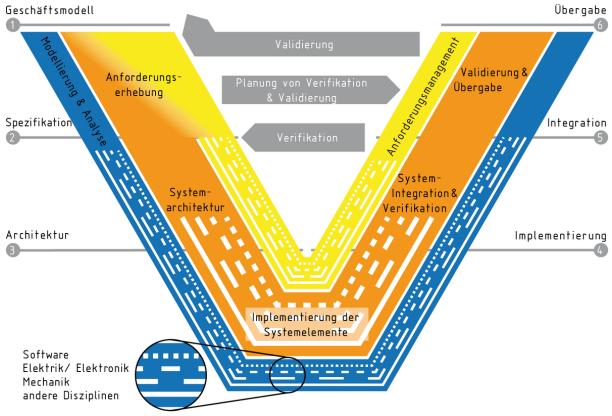


Abbildung 2.5: V-Modell aus der VDI/VDE 2206 (Gräßler et al., 2021)

2.6 Allgemeine Hardware Teststrategien

2.6.1 Modellbasiertes Testen

Wie auch bei den Softwareteststrategien gibt es im Bereich des Hardware-Testings das modellbasierte Testen. Dieses ähnelt in seinen Grundzügen stark dem modellbasierten Testen beim Software-Testing. Dieses wurde in Kapitel 2.5.1 bereits beschrieben.

2.6.2 Manuelles Testen

Da SPS-Programme sowohl kleine als auch große Hardware-Geräte wie Pumpen, Ventile und Generatoren im Feld steuern, ist es fast unmöglich, die Hardware-Geräte physisch vom Feld in die Testumgebung zu bewegen. Die Hardware-Geräte sind in der Regel sehr teuer und können noch in den Produktionssystemen arbeiten, daher ist es sehr teuer, die SPS-Programme in solchen Produktionssystemen zu testen. Es ist ein SPS-I/O-Simulationssystem erforderlich, um die Entwicklung des SPS-Programms während der Entwicklungsphase zu unterstützen und das SPS-Programm in der Testphase zu testen. Ein Hardware-Teststand wird oft verwendet, um die I/O von Hardware-Geräten im Feld zu simulieren. Der Prüfstand besteht aus vielen LED-Anzeigen, Kippschaltern, analogen Signalgeneratoren und Messgeräten, die mit den Eingangs- und Ausgangsmodulen der SPS verdrahtet sind. Die Kippschalter und die analogen Signalgeneratoren werden verwendet, um die Sensoreingänge zu simulieren, und die LED-Anzeigen und die Messgeräte werden verwendet, um die SPS-Ausgänge anzuzeigen (F. Zhang,

2011).

2.7 Interoperabilität in industriellen Anlagen

Die Integration und Vernetzung von Komponenten und Diensten der intelligenten Fertigung innerhalb und über die Grenzen der Fabrik hinaus, erfordern einen nahtlosen Austausch von Informationen mit einer Syntax und Semantik, die von allen beteiligten heterogenen Systemen verstanden wird. Diese Fähigkeit wird als Interoperabilität bezeichnet. Es müssen sich die Produktionsanlagen gegenseitig und die Kontrollsysteme über Prozessaktualisierungen und mögliche Fehler oder Konflikte informieren. Sie müssen gegebenenfalls auch mit Cloud-basierten Plattformen interagieren, um die erforderlichen Dienste zu erhalten. Die Software, auf der diese Komponenten und Dienste aufbauen, muss mit ihrer Umwelt kommunizieren können. Dies kann erreicht werden, indem die Software ein Verständnis für eine gemeinsame Sprache oder Syntax zwischen den anderen Komponenten und Diensten entwickelt (Zeid et al., 2019).

2.7.1 Definitionen der Interoperabilität

Mehrere Definitionen der Interoperabilität wurden von verschiedenen Normungsorganisationen und Institutionen vorgeschlagen. Ganz allgemein kann Interoperabilität als die Fähigkeit zweier oder mehrerer Einheiten zur Interaktion und Zusammenarbeit definiert werden. ISO 16100 definiert Interoperabilität als „die Fähigkeit, Informationen unter Verwendung einer gemeinsamen Syntax und Semantik gemeinsam zu nutzen und auszutauschen, um eine anwendungsspezifische funktionale Beziehung über eine gemeinsame Schnittstelle zu erfüllen“ (ISO 16100-1:2009, 2009). Das IEEE Standard Computer Dictionary definiert Interoperabilität als „die Fähigkeit von zwei oder mehr Systemen oder Komponenten, Informationen auszutauschen und die ausgetauschten Informationen zu nutzen“ (IEEE Std 610.12-1990, 1990). Es gibt viele weitere Definitionen von Interoperabilität, die in unterschiedlichen Zusammenhängen verwendet werden. Die wichtigsten wurden im Folgenden zusammengestellt und aufgelistet.

- Die Fähigkeit eines Systems, von einem anderen System übermittelte Informationen von gegenseitigem Interesse in verständlicher Form zu empfangen und zu verarbeiten (Kasunic, 2001).

- Interoperabilität bedeutet die Fähigkeit von zwei oder mehr Parteien, Maschinen oder Menschen, einen perfekten Austausch von Inhalten durchzuführen. Perfekt bedeutet, dass es keine wahrnehmbaren Verzerrungen oder unbeabsichtigten Verzögerungen zwischen der Entstehung, Verarbeitung und Nutzung von Inhalten gibt (Ford, 1980).

- Interoperabilität zwischen Komponenten großer, verteilter Systeme ist die Fähigkeit, Dienste und Daten miteinander auszutauschen (Heiler, 1995).

- Der Zustand, der zwischen Systemen erreicht wird, wenn Informationen oder Dienste direkt und in zufriedenstellender Weise zwischen den Systemen und/oder ihren Benutzern ausgetauscht werden (D. Raymond und Curts, 1999).

- Die Fähigkeit, Daten, Funktionen und Prozesse in Bezug auf ihre Semantik zu integrieren (Berre et al., 2004).

- Interoperabilität im weiteren Sinne bezieht sich auf den Einsatz computergestützter Werkzeuge, die die Koordinierung der Arbeit und des Informationsflusses über Organisationsgrenzen hinweg erleichtern, wobei der Schwerpunkt auf unternehmensübergreifenden verteilten Geschäftsprozessen und Flüssen liegt (Chituc, Toscano und Azevedo, 2008).

- Die Fähigkeit von Systemen, Einheiten oder Kräften Services anzubieten und von anderen Systemen Dienste anzunehmen und die auf diese Weise ausgetauschten Dienste zu nutzen, damit sie effektiv zusammenarbeiten können (Morris und Levine, 2004).

In der intelligenten Fertigung nimmt die Interoperabilität zwei allgemeine Formen an. Die erste Form entspricht der vertikalen Integration, z. B. der Interoperabilität zwischen der Fertigungssoftware, den Abteilungen in der Fertigung, den von den verschiedenen Anlagen ausgeführten Prozessen, den verschiedenen Fertigungssystemen usw. (Chen und Vernadat, 2002). Die zweite Form entspricht der horizontalen Integration, also der Interoperabilität zwischen intelligenten Automatisierungsgeräten, Cloud-Diensten, Cloud-Plattformen und Unternehmen. Die erfolgreiche Umsetzung der unternehmensweiten Interoperabilität würde zu effektiven und reibungslosen Abläufen in der Fertigungsindustrie führen, wodurch Kosten gesenkt und die Produktions- und Produktqualität erhöht würden.

2.7.2 Herausforderungen der Interoperabilität

Die Faktoren, die sich auf die Interoperabilität auswirken, sind angesichts der Komplexität der Prozesse zwangsläufig multivariat. Das Manufacturing Interoperability Program des NIST (National Institute of Standards and Technology) listet mehrere Faktoren auf, die die Wirksamkeit der Interoperabilität beeinflussen (S. Kemmerer, 2009):

- Übertragung von Daten zwischen Systemen, die ähnlich oder unähnlich sein können (kommerziell).
- Datenübertragung zwischen Software desselben Anbieters (oder Herstellers), die in unterschiedlichen Versionen auf den Systemen vorhanden ist.
- Kompatibilität zwischen verschiedenen Softwareversionen (neuere und ältere Versionen).
- Fehlinterpretation der verwendeten Terminologie oder des Verständnisses der für den Austausch von Daten oder Informationen verwendeten Terminologie.
- Verwendung von nicht standardisierten Unterlagen, auf denen der Datenaustausch verarbeitet oder formatiert wird.
- Das Nichttesten von Anwendungen, die als konform gelten, da es keine Möglichkeiten gibt, dies zwischen den Systemen zu tun.

Weitere Hindernisse für die Interoperabilität sind uneinheitliche Datenformate oder Standards, die Konnektivität im IoT-Bereich und die große Vielfalt der im Handel erhältlichen Produkte.

2.7.3 Implementierung von Interoperabilität

Im Laufe der Jahre gab es viele Ansätze zur erfolgreichen Umsetzung der Interoperabilität. Der IEEE Guide to Enterprise IT Body of Knowledge (EITBOK) hat die Ansätze hauptsächlich in zwei Typen eingeteilt: Syntaktische Interoperabilität und Semantische Interoperabilität (Mosley et al., 2010).

2.7.3.1 Syntaktische Interoperabilität

Das Europäische Institut für Telekommunikationsnormen (ETSI) definiert die syntaktische Interoperabilität wie folgt: „Syntaktische Interoperabilität wird gewöhnlich mit Datenformaten in Verbindung gebracht. Sicherlich müssen die von Kommunikationsprotokollen übertragenen Nachrichten eine genau definierte Syntax und Kodierung haben, auch wenn diese nur in Form von Bittabellen vorliegen. Viele Protokolle übertragen jedoch Daten oder Inhalte, die mit Hilfe von High-Level-Übertragungssyntaxen wie Hypertext Markup Language (HTML), Extensible Markup Language (XML) oder Abstract Syntax Notation One (ASN.1) dargestellt werden können“ (Veer und Wiles, 2008). Syntaktische Interoperabilität berücksichtigt lediglich das Format der Daten, nicht aber die Bedeutung der zu übertragenden Daten. Die Standardisierung der Datenformate und der Kommunikationsmodi kann diese Art der Interoperabilität erheblich verbessern.

2.7.3.2 Semantische Interoperabilität

Das ETSI bietet eine konkrete Definition für semantische Interoperabilität wie folgt: „Semantische Interoperabilität ist in der Regel mit der Bedeutung von Inhalten verbunden und betrifft eher die menschliche als die maschinelle Interpretation der Inhalte. Interoperabilität auf dieser Ebene bedeutet also, dass zwischen Menschen ein gemeinsames Verständnis der Bedeutung der ausgetauschten Inhalte (Informationen) besteht“ (Veer und Wiles, 2008). Die semantische Interoperabilität befasst sich nicht nur mit der Bedeutung des Inhalts, sondern wendet auch eine Logik auf die übertragenen und verwendeten Fakten an (Shukla, Harris und Davies, 2010). XML und Resource Definition Framework (RDF) sind als Standards für semantische Interoperabilität weit verbreitet. Dennoch hat sich RDF als effektiver erwiesen, da es Modelle bereitstellt, die auf mehrere Techniken erweiterbar sind und durch eindeutige Ontologien dargestellt werden (Decker et al., 2000). Eine ausführliche Liste von Ontologien nach den Geräten, die sie verwenden, wurde von Lelli (Lelli, 2019) veröffentlicht, in der das World Wide Web Consortium (W3C) Semantic Sensor Network, Fiesta-IoT, Smart Energy Aware Systems (SEASD), Machine-to-Machine (M2M)-Lösungen und Schema.org-Technologien, um nur einige zu nennen, diskutiert und verglichen werden.

2.7.4 Vertikale Integration

Der Begriff Interoperabilität in der Fertigung bezieht sich auf die Fähigkeit von Fertigungsunternehmen, technische oder unternehmensbezogene Informationen auf kohärente Weise innerhalb und zwischen den Unternehmen auszutauschen (Ray und Jones, 2006). Jones und Ray (Ray und Jones, 2006) schlagen drei Ansätze vor, um das mit der Interoperabilität in der Fertigung verbundene hohe Kostenproblem in den Griff zu bekommen. Der erste ist die M2M-Lösung. Die Idee hinter diesem Ansatz ist, jede einzelne Maschine mit jeder anderen Maschine, mit der sie verbunden oder integriert ist, interoperabel zu machen. Die Herausforderung besteht darin, dass jede dieser Maschinen auf der Grundlage ihres eigenen, vom Hersteller spezifizierten Kommunikationsprotokolls, kommunizieren kann. Um in diesem Szenario Interoperabilität zu erreichen, ist ein gründliches Verständnis der einzigartigen Semantik dieser Geräte sowie die Übersetzung ihrer Syntax erforderlich. Dieser Ansatz ist eindeutig nicht effektiv und wird wahrscheinlich erhebliche Kosten verursachen.

Der zweite Ansatz ist eine branchenweite Standardisierungslösung. Hier soll sichergestellt werden, dass alle Dienstleistungspartner in der Fertigungsindustrie einer einzigen Lösung folgen. So kann beispielsweise eine Fertigungseinheit bis zur Produktionsphase integriert und interoperabel sein. Wenn jedoch in späteren Phasen weitere Prozesse am Produkt durchgeführt werden müssen, kann es sein, dass es in ein anderes Werk verlagert werden muss, um diese Vorgänge abzuschließen. In diesem Fall ist nicht nur ein gemeinsames Protokoll für die Vorgänge innerhalb der Produktionsanlage erforderlich, sondern auch ein weiteres Protokoll zwischen den verschiedenen Produktionseinheiten. Dieser Ansatz kann auf verschiedene Branchen ausgeweitet werden. Der Nachteil dieses Ansatzes ist jedoch, dass die Produktionseinheiten, die ein bestimmtes Produkt verarbeiten, zu verschiedenen Unternehmen mit unterschiedlichen Integrations-/Kommunikationsprotokollen gehören können. Der dritte und effektivste Ansatz ist die Anwendung offener Standards oder Plattformen, um Interoperabilität zu erreichen. Eine Übersicht über die Entwicklungen im Bereich der offenen Standards zur Interoperabilität findet sich in (Hofman, 2019) und (Chen, Doumeingts und Vernadat, 2008).

2.7.5 Horizontale Integration

Die Norm ISO/IEC 19941 (ISO/IEC 19941:2017, 2017) spezifiziert ein facettenbasiertes Modell, um sowohl Interoperabilität als auch Portabilität im Cloud Computing zu erreichen. Wenn man den Begriff des Cloud Computing auf die Cloud-Fertigung (Xu, 2012) ausweitet, ist es zwingend erforderlich, die Interoperabilitätsdefinitionen und -ansätze auf den Bereich der Cloud-Fertigung zu verallgemeinern. Zusätzlich zur syntaktischen und semantischen Interoperabilität müssen Cloud-basierte Plattformen drei weitere Arten der Interoperabilität berücksichtigen:

Transport-Interoperabilität

Dieser Typ macht die Übertragung und den Austausch von Daten interoperabel durch die Verwendung von Protokollen wie Representational State Transfer (REST) (Fielding, 2000) über HyperText Transfer Protocol (HTTP) und Message Queuing Telemetry Transport (MQTT) (ISO/IEC 20922:2016, 2016).

Verhaltensbasierte Interoperabilität

Dieser Typ wählt aus einer Liste erwarteter Antworten aus, wenn Anfragen für unsere Dienste eingehen. Durch die Festlegung von Bedingungen für die Anfragen können Computersimulationen mit Hilfe menschlicher Antworten die Ergebnisse der Dienstanfrage vorhersehen und eine Lösung anbieten.

Interoperabilität der Politik

Diese Art von Interoperabilität stellt sicher, dass alle Systeme in der Cloud bei der Interaktion innerhalb der Cloud-Umgebung die Vorschriften und Richtlinien befolgen und einhalten.

In der Cloud-Fertigung werden die meisten Vorgänge und Ressourcen auf Cloud-Plattformen als Cloud-basierte Dienste virtualisiert, die durch IoT, Cloud Computing und cyber-physische Technologien unterstützt werden. In den letzten Jahren sind mehrere cloudbasierte Plattformen entstanden, die solche cloudbasierten Fertigungsdienste anbieten. Zu den gängigen Plattformen gehören unter anderem Amazon Web Services (AWS), IBM Watson,

Microsoft Azure, Google Cloud Platform, Oracle und Alibaba. Die Nutzer dieser Dienste können Dienste aus der Ferne und auf Abruf von mehreren Anbietern für ihr Fertigungsunternehmen abonnieren und nutzen. Zu den angebotenen Diensten gehören unter anderem Datenbankmanagement, Big-Data-Analysen, Terminplanung und Supply-Chain-Operationen.

3 Anforderungsanalyse

3.1 Adaption von Strategien der VIBN und CS

Grundlegend für die Analyse der Anforderungen für die Kompatibilitätsprüfung war die theoretische Betrachtung der bereits existierenden Strategien der VIBN und der CS. Diese wurden im Stand der Technik dargelegt und bieten einen guten Ausgangspunkt, um Ideen aus diesen zwei Bereichen für die Entwicklung eines Testkonzeptes zur Kompatibilitätsprüfung zu nutzen.

Aus TA 4 folgt zunächst die Bestimmung von Kriterien für die Teststrategien aus der VIBN und der CS, um als wesentlich eingestuft zu werden. Das Hauptkriterium für die Teststrategien, um auch für eine Kompatibilitätsprüfung in Frage zu kommen ist das Vorhandensein eines Modells auf das geprüft wird. Da nur zwei der im Stand der Technik vorgestellten Teststrategien diesem Kriterium entsprechen, werden diese als wesentlich für die Kompatibilitätsprüfung eingestuft.

Besonders die Strategien SiL und HiL der VIBN (vgl. Kap. 2.1.1) zeigen, dass die Verwendung eines vorher bestimmten Modells enorm wichtig ist für den ordnungsgemäßen Ablauf eines Tests. Aus dem Bereich der CS existiert die Strategie der anomaliebasierten Erkennung (Kap. 2.2.2.2) eines Angriffs durch die Nutzung von Markov-Modellen. Wie von Ourston et al. (Ourston et al., 2003) geschildert, beinhaltet diese ebenfalls ein vorher bestimmtes Modell, um das Auftreten von Abweichungen und damit einen potenziellen Angriff zu erkennen.

Abgeleitet von dieser Erkenntnis ist demnach auch für die Kompatibilitätsprüfung ein Modell von großer Bedeutung. Im Weiteren wird dieses Modell auch als Soll-Zustand bezeichnet, welches die zur Prüfung der Kompatibilität vorgesehene Software und Hardware beinhaltet.

Dem Soll-Zustand gegenüber steht, wie bei den Strategien SiL oder HiL, das zu testende System, bzw. die simulierte oder physische SPS auf die die Prüfung angewendet werden soll. Bei der Kompatibilitätsprüfung wird dieses zu testende System im nachfolgenden als IST-Zustand bezeichnet. Dieser bildet den aktuellen Zustand der mit der SPS verbundenen Hardwaregeräte sowie den Zustand der Software dieser Geräte ab.

Damit stehen die beiden Hauptpunkte für das Konzept der Kompatibilitätsprüfung fest, welche die Bestimmung eines SOLL- und IST-Zustandes umfassen.

Übertragbarkeit der Strategien der VIBN und CS auf Software-Upload und Software-Neustart

Die aus der VIBN und CS abgeleiteten Strategien der Modellprüfung und der anomaliebasierten Erkennung lassen sich durch Hinzunahme einer weiteren Komponente außerhalb der SPS auf Upload und Neustart übertragen. Diese weitere Komponente übernimmt dann das Überwachen der SPS und löst automatisiert eine Kompatibilitätsprüfung aus, sobald ein Update ansteht oder die SPS neugestartet wurde. Im vorgeschlagenen Konzept (siehe Kap. 4) ist diese Komponente ein Industrie PC (I-PC). Auf diesem PC läuft ein Testskript, welches den Software-Upload auf die SPS steuert und den Software-Neustart der SPS überwacht.

3.2 Anforderungen

Ausgehend von der Aufgabenstellung und den formulierten Teilaufgaben ergeben sich folgende Anforderungen an die Komponenten der Kompatibilitätsprüfung.

3.2.1 Anforderungen an die Kompatibilitätsprüfung

Diese Anforderungen bilden die Grundlage für die Entwicklung des Konzeptes für die Kompatibilitätsprüfung.

A. Die Kompatibilitätsprüfung muss eine Aussage über die Kompatibilität, also die Verträglichkeit des simultanen Betriebs der mit der SPS verbundenen Hardware und Softwarekomponenten treffen und diese dem Testingenieur anzeigen.

B. Die Kompatibilitätsprüfung muss automatisiert während dem SPS-Neustart bzw. während des Aufspielens von Software auf die SPS ablaufen können.

C. Das Ergebnis der Kompatibilitätsprüfung muss eine Aussage über gefundene Inkompatibilitäten und Fehler ermöglichen.

D. Am Ende der Kompatibilitätsprüfung sollen dem Testingenieur angemessene Reaktionen auf gefundene Inkompatibilitäten und Fehler aufgezeigt werden.

E. Die Kompatibilitätsprüfung soll mit Hardware und Software verschiedener Hersteller der Industrie funktionieren.

3.2.2 Anforderungen an die Hardwarekomponenten

Um die Durchführung einer Kompatibilitätsprüfung zu gewährleisten, müssen die zu testenden Hardwarekomponenten bestimmten Anforderungen genügen. Diese Anforderungen werden in diesem Abschnitt beleuchtet.

A. Die zu testende Hardwarekomponente muss physisch oder drahtlos mit dem I-PC, auf dem die Kompatibilitätsprüfung durchgeführt wird, verbunden sein. Dies kann entweder durch eine drahtgebundene oder drahtlose Netzwerktechnologie erfolgen.

B. Der ethernet-basierte Kommunikationsstandard Profinet I/O muss von der Hardwarekomponente unterstützt werden.

C. Die Komponente muss das Profinet Discovery and Configuration Protocol (DCP) (vgl. (PROFINET University, 2018)) unterstützen.

D. Die auf der Hardwarekomponente installierten Bibliotheken müssen auslesbar sein.

E. Die installierte Firmware muss auslesbar sein.

3.2.3 Anforderungen an die Softwarekomponenten

Ähnlich wie die Hardwarekomponenten müssen auch die zu testenden Softwarekomponenten bestimmten Anforderungen genügen. Diese Anforderungen werden nachfolgend aufgezeigt.

A. Die Software muss in dem maschinenlesbaren Format von PLCopen XML oder CAEX vorliegen.

B. Die Steuerungslogik und Hardwarekonfiguration aus der Software muss frei von Logikfehlern und Widersprüchen sein.

C. Die PLCopen XML oder CAEX Datei darf nicht beschädigt sein und muss dem jeweiligen Standard entsprechen.

4 Konzept

Die in diesem Kapitel behandelte Synthese von Erwartungen und Erfahrungen bezüglich der Erstellung einer Teststrategie bildet die Präskriptive Studie (Biedermann et al., 2012). Die Erarbeitung eines Konzeptes für die Kompatibilitätstests und damit die Durchführung der ersten anfänglichen präskriptiven Forschung wird in diesem Kapitel vorgestellt. Ausgehend von den Forschungsfragen und den Teilaufgaben TA10 und TA11 wird der Lösungsansatz erstellt.

Das vorgeschlagene Konzept besteht aus vier wichtigen Teilabschnitten. Diese Teilabschnitte werden in den folgenden Unterkapiteln näher beschrieben.

Wie im Kapitel zur Anforderungsanalyse bereits beschrieben, sollen im Konzept der Soll- und Ist-Zustand der Software und Hardwarekomponenten essentieller Bestandteil des Kompatibilitätstest sein. Ein stringenter Testablauf, der immer genau gleich wiederholt werden kann, bildet den Rahmen für das Konzept und wird im nächsten Unterkapitel vorgestellt.

4.1 Komponenten des Teststrategiekonzeptes

Die Durchführung des Kompatibilitätstest erfolgt in vier Phasen (siehe Abb. 4.1). In der ersten Phase wird auf der SPS, welche mit einem Test-I-PC verbunden ist, ein automatisierter Selbsttest durchgeführt. Dieser umfasst je nach Hersteller verschiedene Testkategorien.

Der SPS Selbsttest soll sicherstellen, dass alle Grundvoraussetzung für den Betrieb der SPS bestehen, welche völlig unabhängig von angeschlossenen Geräten oder Software sind.

Diese Grundvoraussetzungen umfassen das Vorhandensein einer CPU und I/O Module, sowie die Selbstüberprüfung der SPS auf fehlenden oder fehlerhaften Speicher und die Sicherstellung, dass genügend Energie am Stromanschluss vorhanden ist (vgl. Kap. 4.1.1).

Phasen zwei und drei umfassen die Ermittlung des Soll- und Ist-Zustandes der Software- und Hardwarekomponenten, welche Teil der Kompatibilitätsprüfung sein sollen. Die Reihenfolge der Ermittlung ist dabei nicht unerheblich. Die Ermittlung des Soll-Zustandes sollte nach der Ermittlung

des Ist-Zustandes stattfinden. So kann nach einer erfolgreichen Kompatibilitätsprüfung direkt das Software-Update aus dem bereits geladenen Soll-Zustand auf die SPS übertragen werden. Die Ermittlung des Soll-Zustandes wird in Unterkapitel 4.1.2 und die Ermittlung des Ist-Zustandes wird in Unterkapitel 4.1.3 genau erläutert.

In der vierten und letzten Phase des Kompatibilitätstest werden die ermittelten Soll- und Ist-Zustände der Software- und der Hardwarekomponenten verglichen und dabei die Unterschiede aufgestellt. Abgeleitet von den gefundenen Unterschieden sollen dann die Inkompatibilitäten ermittelt werden und in bestimmte Kategorien der Fehlerdetektionstabellen eingeordnet werden (vgl. Kap. 4.2).

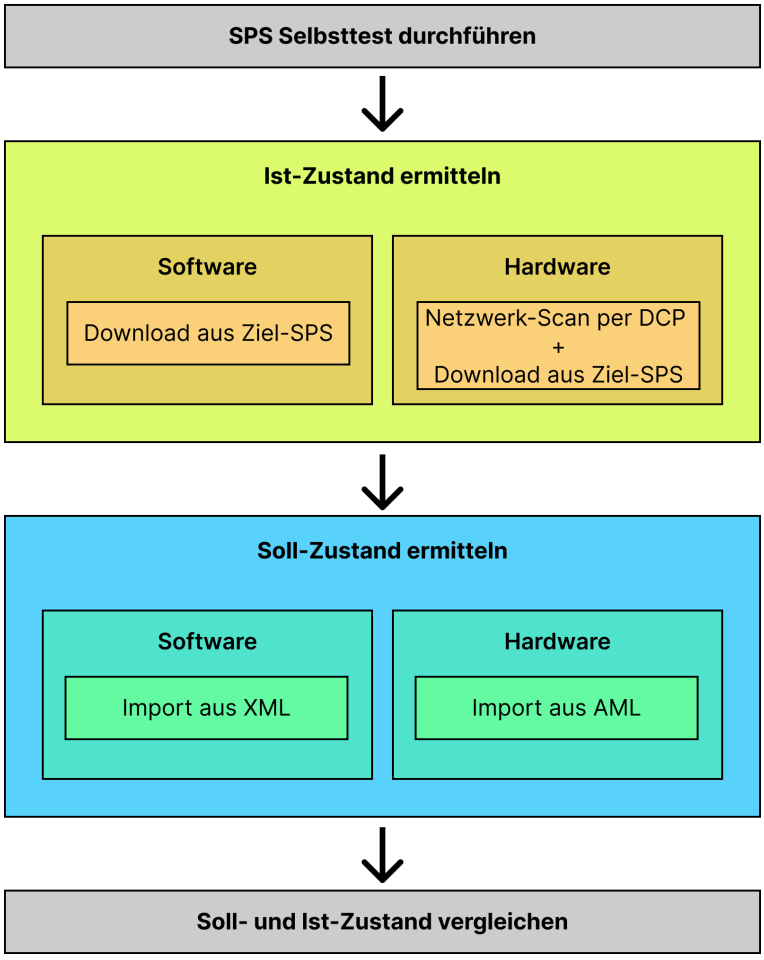


Abbildung 4.1: Ablauf der Kompatibilitätsprüfung

4.1.1 SPS Selbsttest

Der SPS Selbsttest umfasst die Selbstkontrolle und die Diagnose einer zu untersuchenden SPS. Wie in der DIN EN 61131-2 gefordert, müssen Hersteller von SPS-Systemen Mittel für Selbsttests und Diagnose der Arbeitsweise dieser Systeme zur Verfügung stellen. Weiterhin muss der Selbsttest eine Aussage über den ordnungsgemäßen Zustand eines SPS-Systems ermöglichen.

Der SPS-Selbsttest nach DIN EN 61131-2 muss Diagnosemittel bereitstellen, um die folgenden Aktionen durchzuführen:

1. Überwachung des Anwendungsprogrammes (watch dogs);
2. Überprüfung der Integrität (Fehlerfreiheit) des Speichers;
3. Überprüfung der Richtigkeit der Daten, welche zwischen Speicher, Verarbeitungseinheit und I/O- Modulen ausgetauscht werden;
4. Überprüfung der Stromversorgung des Systems;
5. Überwachung des Zustandes der Hauptverarbeitungseinheit MPU.

Der Ausgang der SPS Selbsttests ist dabei essentiell um die Eignung der SPS für die Kompatibilitätsprüfung festzustellen. Mit dem Selbsttest wird die Grundlage dafür geschaffen, den Hardwareanforderungen des Kompatibilitätstests zu genügen. Ein System, welches sich nicht im ordnungsgemäßen Zustand befindet, also welches den Selbsttest nicht besteht, kann nicht als Teil des Konzeptes zur Prüfung der Kompatibilität zwischen Software und Hardware eingesetzt werden.

4.1.2 Export & Import des Soll-Zustandes

Bevor der Soll-Zustand zur Überprüfung der Kompatibilität mit der neuen Hardware importiert werden kann, müssen zunächst die für den Import benötigten Dateien beschafft werden. Je nach Anwendungsfall unterscheiden sich die Anforderungen für den Import des Soll-Zustandes.

Anwendungsfälle

A. Die Kompatibilitätsprüfung soll während des Neustarts der SPS ablaufen.

B. Die Kompatibilitätsprüfung soll vor dem Aufspielen neuer Software auf die SPS erfolgen.

Anwendungsfall A Beim Import des Soll-Zustandes beim Anwendungsfall A kommt der Soll-Zustand direkt aus zuvor gespeicherten Daten auf dem Test-PC. Dieser letzte Zustand des Gesamtsystems stellt den letzten bekannten Ist-Zustand vor dem Neustart der SPS dar. Damit ein automatisierter Prüfablauf garantiert werden kann, speichert der Test-PC jeweils den letzten ermittelten Ist-Zustand der SPS und hält diesen für die Kompatibilitätsprüfung im Falle des Neustarts der SPS bereit.

Durch einen ständigen Ping zwischen Testskript und SPS wird erkannt, wann es zu einem Neustart der Ziel-SPS kommt. Sobald ein Neustart erkannt worden ist, wird automatisch die Kompatibilitätsprüfung eingeleitet. In Abb. 4.2 kann man die Komponenten, welche bei Anwendungsfall A zu berücksichtigen sind, erkennen.

Da der Soll-Zustand zum Zeitpunkt eines Neustarts bereits auf dem I-PC vorhanden ist, muss dieser nur noch durch das Testskript geladen werden.

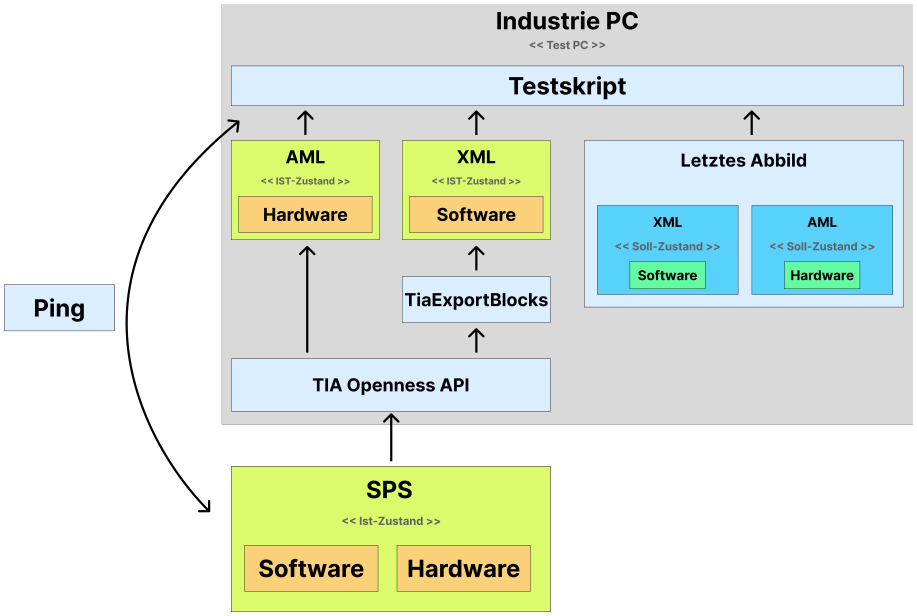


Abbildung 4.2: Ermittlung des Soll- und Ist-Zustandes beim Neustart

Anwendungsfall B Beim Aufspielen neuer Software auf die SPS kommt der Soll-Zustand aus dem durch den Anwender bereitgestellten Updatedaten. Sollten die Updatedaten aus einer TIA-Projektdatei bestehen, wird diese Datei zunächst durch die TIA Openness Api geöffnet. Anschließend wird die

Open-Source Software TiaExportBlocks (Suteu, 2020) genutzt, um die Variablentabellen im XML-Format aus dem TIA Projekt zu extrahieren und in eine XML-Datei zu exportieren. Mithilfe der CAx-Export Funktion der TIA Openness API wird die Hardwaretopologie als AutomationML(AML)-Datei (AutomationML, 2022) extrahiert. Die Daten der Software liegen dann als XML-Datei und die Daten der Hardware als AML-Datei vor. Die AML-Datei bedient sich dabei dem CAEX-Format. und können durch das Testskript eingelesen werden (AutomationML, 2022). In Abbildung 4.3 ist dieser Prozess auf der rechten Seite dargestellt.

Nun liegt dem Testskript der für die Kompatibilitätsprüfung erforderliche Soll-Zustand der Software und Hardware vor.

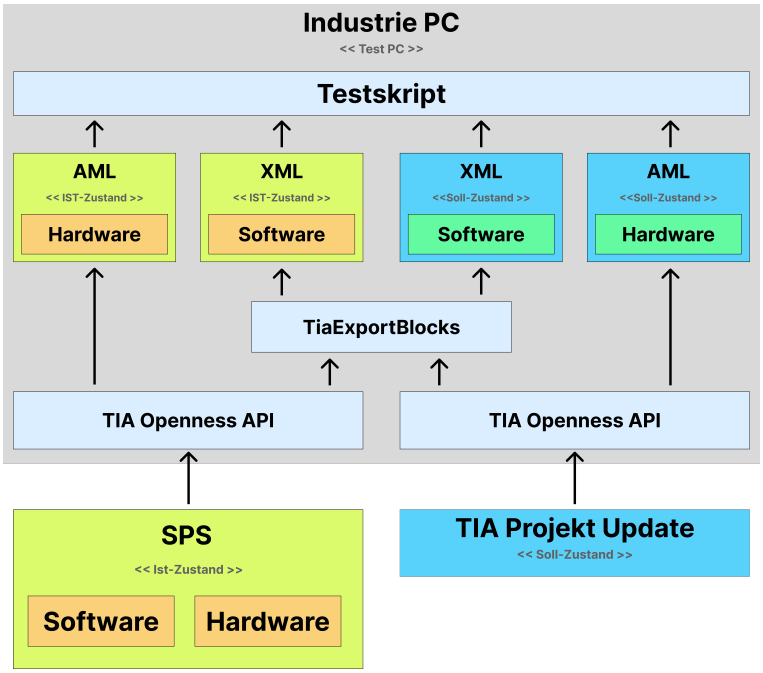


Abbildung 4.3: Ermittlung des Soll- und Ist-Zustandes beim Software Update

Das CAEX Dachformat ist definiert in der IEC 62424 Norm (IEC 62424, 2016)und benutzt den PLCopen XML Standard (PLCopen foundation, 2009), um die Hardwaretopologie maschinenlesbar zu machen.

Durch die Ermittlung des Soll-Zustands werden die nötigen Hardware- und Softwareinformationen gesammelt, um diese dann im nächsten Schritt mit den gesammelten Informationen des Ist-Zustandes zu vergleichen.

4.1.3 Ermittlung des Ist-Zustandes

Bei der Ermittlung des Ist-Zustandes der Software und Hardwarekomponenten wird der aktuelle Zustand des Netzwerkes durch den DCP Scan ermittelt und anschließend die SPS mithilfe der TIA-Portal Openness API von Siemens (Siemens Openness Api Handbuch, 2021) programmatisch durch den I-PC kontaktiert und ausgelesen. Dieser Ermittlungsprozess ist in Abbildungen 4.3 und 4.2 jeweils unterhalb des I-PC dargestellt.

Dafür wird zunächst durch das Siemens Automation Tool (SAT) ein DCP-„Identify All“ Befehl an das Ethernet Netzwerk ausgesendet, welches mit dem Test PC verbunden ist. Der Ablauf des DCP Befehls ist in Abbildung 4.4 gezeigt und veranschaulicht den Hardwaredetektionsprozess. Um PROFINET Geräte zu finden muss diese physisch mit dem Netzwerk verbunden sein. Der DCP-Befehl findet alle PROFINET (PROFINET University, 2018) Geräte, welche das DCP-Protokoll unterstützen, physisch mit dem Netzwerk verbunden sind sowie vom Test-PC aus erreichbar sind. Wenn ein Gerät gefunden wurde, liefert der DCP-„GET“ Befehl die Hardwareinformationen des Geräts wie zum Beispiel den PROFINET Gerätename, IP Adresse, Firmwareversion und MAC Adresse. Durch das programmatische Hinzufügen der gefundenen Geräte in eine Liste ist es später auch noch möglich, weitere Informationen über gefundene Geräte abzurufen (siehe S. 227 zu IProfinetDeviceCollection (Siemens SAT Handbuch, 2021)). Diese Zusatzinformationen sind aber nicht weiter für das vorgestellte Konzept von Relevanz.

Anschließend wird die SPS Software durch die TIA Openness API auf den I-PC heruntergeladen. Dabei kann nach erfolgreichem Herunterladen, durch die Auswahl von „CAx-Daten exportieren“ aus dem Reiter „Werkzeuge“, die AML Datei für die Hardwarekonfiguration erzeugt werden(siehe Abb. 5.5).

Die im SPS Code vorhandenen Variablen können mittels der Open-Source Software „TiaExportBlocks“ aus einem geöffneten Projekt automatisiert zu maschinenlesbaren XML-Dateien exportiert werden.

4.1.4 Vergleich des Soll- & Ist-Zustandes

Das Kernelement des Konzeptes ist der Vergleich des Soll- und Ist-Zustandes, der aus den vorangegangenen Kapiteln hervorgeht. Dabei werden die Unterschiede ermittelt und mögliche Probleme aufgedeckt.

Bei dem Blick auf die Unterschiede kann man Inkompatibilitäten zwischen den Hardware- und Softwarekomponenten erkennen und in eine Tabelle (vgl. Kapitel 4.2) einordnen und kategorisieren. Dieser Prozess wird automati-

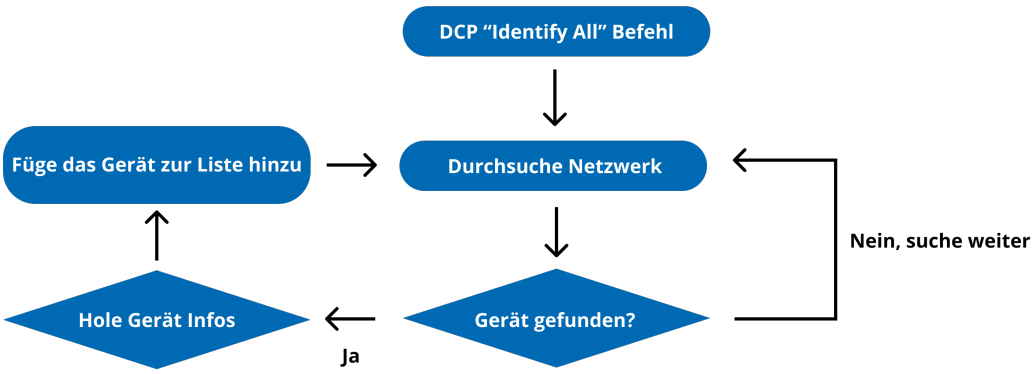


Abbildung 4.4: Ablauf des DCP-Scans, nach (PROFINET University, 2018)

siert durch das Testskript auf dem I-PC durchgeführt. Dieser automatisierte Test-Prozess kann beim Hochladen neuer Software oder beim Neustart des Systems durchgeführt werden.

4.2 Fehlerdetektionstabellen

Nachfolgend aufgezeigt sind die Fehlerdetektionstabellen, welche als Ergebnis der Kompatibilitätsprüfung automatisiert durch das Testskript erstellt werden. Diese sind aufgeteilt nach Komponenten des Testablaufes, wie in Abbildung 4.1 dargestellt. Mögliche Reaktionen auf die gefundenen Inkompatibilitäten sind ebenfalls in den Tabellen mit angegeben. Zur Vereinfachung der Schreibweisen innerhalb der Tabellen werden folgende Abkürzungen eingeführt:

- Hardware (**H/W**)
- Software (**S/W**)
- Nicht vorhanden (**n.v.**)
- Lokal lösbares Problem (**L**)
- Remote lösbares Problem (**R**)

Tabelle 4.1: Fehlerdetektionstabelle für SPS Selbsttest

SPS Selbsttest	
Erkennbarer Fehler	Reaktion
CPU, I/O Module n.v.	Hardware vor Ort reparieren, einbauen, austauschen(L)
Fehler im Anwendungsprogramm	Code überprüfen und Updaten(R)
Datenaustausch fehlerhaft	SPS überprüfen(L)
Speicherintegrität verletzt	Speicher prüfen(L)
Komm. Schnittstelle n.v.	Stromversorgung prüfen(L)

Tabelle 4.2: Fehlerdetektionstabelle für Import & Export des Soll-Zustandes

Import & Export des Soll-Zustandes	
Erkennbarer Fehler	Reaktion
Fehler in der Logik des SPS-Code	Logikfehler im Code beheben(R)
H/W-Topologie n.v.	Dateistrukturen prüfen(R)
SPS-Code n.v.	Dateistrukturen prüfen(R)
Datei-/XML-Struktur fehlerhaft	AML neu exportieren+importieren(R)

Tabelle 4.3: Fehlerdetektionstabelle für Ermittlung des Software Ist-Zustandes

Ermittlung des Software Ist-Zustandes	
Erkennbarer Fehler	Reaktion
SPS Code n.v.	Verbindung zur SPS prüfen(L)
SPS-System n.v.	Verbindung zur SPS prüfen(L)
SPS in falschem Netzwerk	Netzwerkkonfiguration prüfen(R)

Tabelle 4.4: Fehlerdetektionstabelle für Ermittlung des Hardware Ist-Zustandes

Ermittlung des Hardware Ist-Zustandes	
Erkennbarer Fehler	Reaktion
Keine H/W-Geräte gefunden	Überprüfe Anschlüsse(L)
DCP Scan erfolglos	Prüfen ob DCP Protokoll unterstützt(L)
Daten aus Gerät nicht abrufbar	Überprüfe Anschlüsse(L)
H/W-Gerät in falschem Netzwerk	Netzwerkkonfiguration prüfen(R)

Tabelle 4.5: Fehlerdetektionstabelle für Vergleich Soll und Ist-Zustand

Vergleich Soll und Ist-Zustand	
Erkennbarer Fehler	Reaktion
IP-Adressen verschieden	IP-Adressen richtig setzen(R)
H/W-Bibliotheken verschieden	Bibliotheken synchronisieren(R)
H/W-Firmware verschieden	Firmware updaten(R)
Speicher nicht ausreichend	Speicher erweitern(L)
H/W-Gerät nicht synchron	Geräteverbindungen prüfen(L)
Code/Variablen referenziert n.v. Geräte	Geräteverbindungen prüfen(L)

4.3 Reaktionen auf Inkompatibilitäten

Die unter Kap. 4.2 genannten Reaktionen bilden eine Möglichkeit für den Testingenieur auf gefundene Inkompatibilitäten adäquat zu reagieren. Dabei muss erwähnt werden, dass die Reaktionen nicht automatisch durch das System erfolgen, was den Eingriff durch den Testingenieur erfordert. Zur Beantwortung der Teilaufgabe TA 10 besteht das Konzept zur Reaktion auf gefundene Inkompatibilitäten daraus, zu identifizieren, an welchen Stellen man reagieren muss, damit die Inkompatibilitäten behoben werden könnten. In den Fehlerdetektionstabellen kann man ebenfalls anhand der in Klammern stehenden Buchstaben (R) und (L) erkennen, ob eine bestimmte Inkompatibilität von außen, also „remote (R) “ behebbar ist oder ob ein Ingenieur vor Ort, also lokal (L) am System einen Eingriff vornehmen muss, um den Fehler zu beheben.

5 Evaluation

Die in diesem Kapitel durchgeführte Evaluation bildet zusammen mit der nachfolgenden Diskussion und Bewertung(vgl. Kap. 6) den Einstieg in die zweite deskriptiven Studie und ist die letzte Stufe der Forschung nach (Biedermann et al., 2012).

Um die in Kapitel 4 beschriebene Kompatibilitätsprüfung für industrielle Steuerungen zu evaluieren und testen zu können, wurden bestimmte Rahmenbedingungen sowie Werkzeuge festgelegt. Diese sind für die Evaluierung des Konzeptes erforderlich.

5.1 Methodik und Evaluationskriterien

Nachfolgend soll explizit das TIA-Portal, sowie weitere Softwarewerkzeuge von Siemens dazu genutzt werden, um beispielhaft an einer Referenzanlage der Firma Sartorius das Strategiekonzept zu evaluieren. Als Methode der Evaluation soll ein Test des Konzepts an einer realen Anlage eingesetzt werden. Die Kriterien zur Evaluation entsprechen den Anforderungen an die Kompatibilitätsprüfung aus Kapitel 3.2.1.

Die Evaluation wird durchgeführt, indem an einem Referenzsystem das Aufspielen einer neuen Software auf eine Hardware SPS getestet wird. Dabei wird das vorgestellte Konzept zur Kompatibilitätsprüfung angewendet und durchgeführt.

5.2 Anwendungsbeispiel

Als mögliches Anwendungsbeispiel für die Kompatibilitätsprüfung bieten sich die industriellen Steuerungssysteme der Firma Sartorius an. Dabei existieren zwei verschiedene Modelle der Steuerung bei Sartorius Anlagen.

Eines der Modelle beinhaltet eine sogenannte Software SPS (siehe Abb. 5.1). Dabei wird auf dem I-PC eine virtuelle Instanz einer vollumfänglich konfigurierbaren und nutzbaren SPS erstellt. Die Kommunikation mit den Feldgeräten wird dann wiederum über ein an den I-PC angeschlossenes I/O-Modul realisiert. Sartorius nutzt in seinen Systemen Automation Skripte,

welche die Bearbeitung von Updates leichter erlauben. WinCC OA wird zusätzlich eingesetzt, um ein Human-Machine-Interface zu implementieren.

Das zweite Modell nutzt eine reguläre Hardware SPS (siehe Abb. 5.2). Diese wird per externer Stromversorgung versorgt und per Ethernet mit dem I-PC verbunden, um das Programmieren der SPS zu ermöglichen. Die Hardware SPS ist dabei physikalisch über Digitale und Analoge I/O Module mit den Feldgeräten verbunden.

Die bereits bestehenden Systeme werden von Sartorius mit dem TIA-Portal von Siemens entwickelt und eignen sich daher, um die Anwendung des Konzeptes zu veranschaulichen. Um das Konzept vor Ort evaluieren zu können, konnte jedoch kein original Sartorius Steuerungssystem genutzt werden. Daher wurden die einzelnen Schritte der Kompatibilitätsprüfung an einem System des P2O-Lab der TU Dresden evaluiert. Das System im P2O-Lab entspricht dem Sartorius Modell mit Hardware SPS.

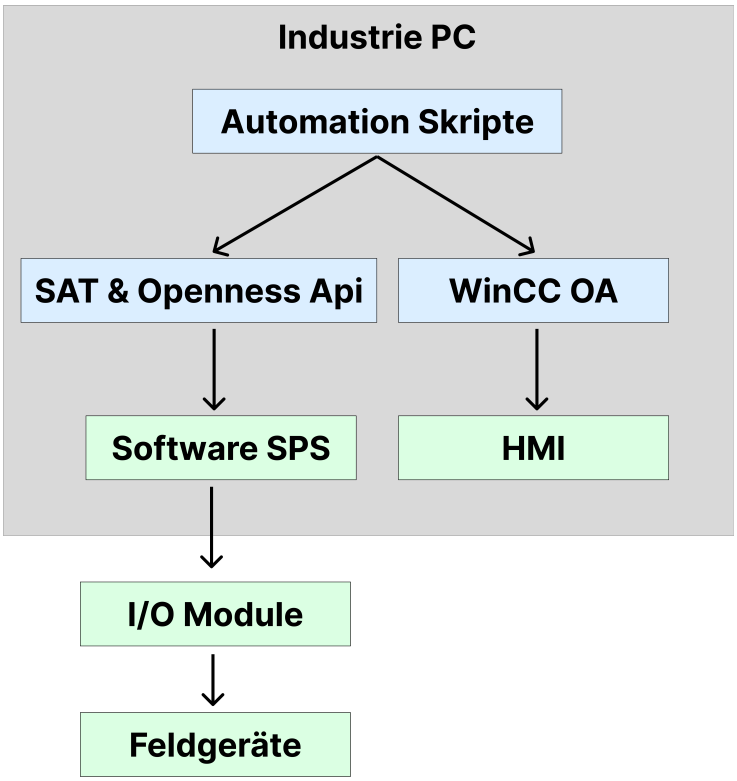


Abbildung 5.1: Modell eines Steuerungsystems von Sartorius mit Software SPS

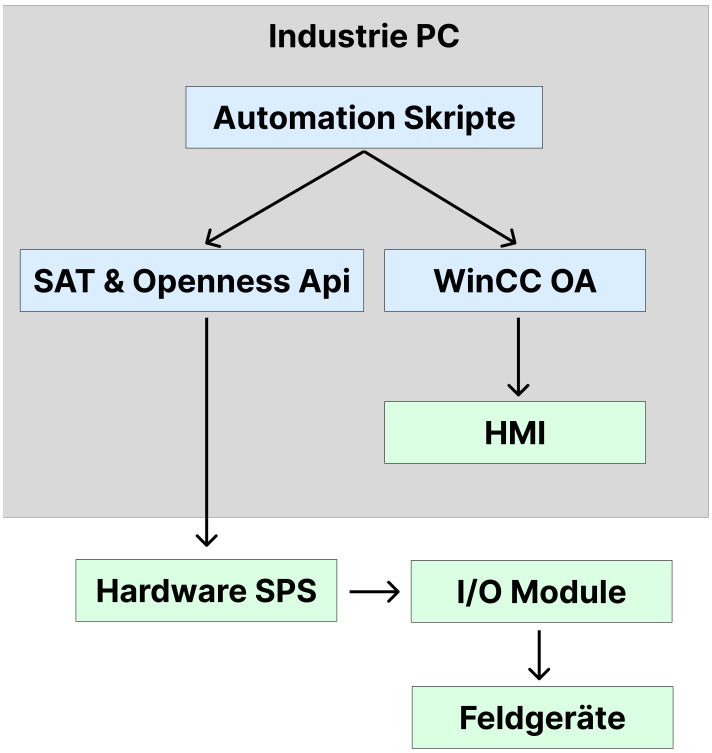


Abbildung 5.2: Modell eines Steuerungsystems von Sartorius mit Hardware SPS

5.3 Referenzsystem für Evaluation

Als Referenzsystem, um die Schritte der Kompatibilitätsprüfung zu evaluieren, wurde ein Steuerungssystem mit Hardware SPS des P2O-Lab der TU Dresden genutzt. Das Referenzsystem entsprach dabei der Architektur des in Abbildung 5.2 gezeigten Modell mit Hardware SPS.

5.4 Durchführung Evaluation

Einschalten der SPS und SPS Selbsttest

Beim Hochfahren der SPS führte diese selbstständig den Test durch und zeigte keinerlei Anzeichen an Fehlern. Der Erste Schritt der Kompatibilitätsprüfung wurde damit erfolgreich abgeschlossen. Wären Fehler aufgetreten, dann hätte man diese am Gerät selbst durch LED-Indikatoren ablesen können. Weiterhin wurde das SPS System per Ethernet mit demselben Netz

verbunden, in dem sich der I-PC befindet.

Vorbereitung I-PC

Um das Konzept im Referenzsystem anzuwenden, wurde zunächst der I-PC auf die Durchführung der automatisierten Kompatibilitätsprüfung vorbereitet.

Installation Zunächst musste das TIA-Portal von Siemens installiert werden. Weiterhin wurde das Open-Source Programm „TiaExportBlocks“ gemäß der Installationsanweisung installiert.

Einrichtung Testskript An dieser Stelle wäre das Testskript eingerichtet worden, damit es ständig im Hintergrund auf dem I-PC läuft. Da dieses Testskript nicht Teil dieser Arbeit ist, wurde dieser Schritt nicht durchgeführt.

Konnektivitätsprüfung Die Verbindung zwischen I-PC und SPS wurde durch einen Ping überprüft.

Ist-Zustand ermitteln

In der zweiten Phase der Kompatibilitätsprüfung wurde der Ist-Zustand des Gesamtsystems erfasst. Dabei wurde der bestehende Code in der SPS analysiert und die Hardwarekonfiguration geladen

Software Ist-Zustand ermitteln Mithilfe des TIA-Portals wurde, wie in Abbildung 5.3 gezeigt, die Software der aktuell verbundenen SPS heruntergeladen. Dies funktioniert nur, wenn ein neues TIA-Projekt vorher generiert wurde. Eine detaillierte Anleitung, wie dieses Herunterladen von SPS-Variablen in das Programmiergerät exakt funktioniert, findet sich auf der Website von Siemens (Siemens, 2018).

Nachdem die Daten auf den I-PC heruntergeladen wurden, mussten diese noch zu XML exportiert und maschinell lesbar gemacht werden. Hierzu wurde die Open-Source Software „TiaExportBlocks“ genutzt (siehe Abb. 5.4).

Diese Ermittlung funktioniert natürlich auch voll automatisiert mithilfe des Testskriptes, welches die Tia Openness Api nutzt, um auf dieselben Funktionen wie das TIA-Portal zuzugreifen.

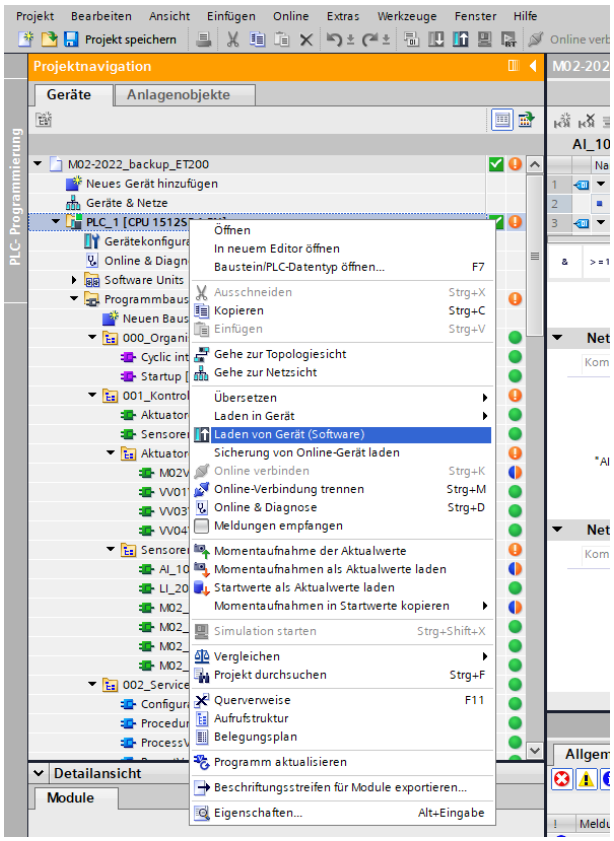


Abbildung 5.3: Software Ist-Zustand der SPS ermitteln

Hardware Ist-Zustand ermitteln Nachdem die Software aus der SPS erfolgreich heruntergeladen worden ist, konnte auch die in den TIA Projektdaten vorhandene Hardwarekonfiguration exportiert und ausgelesen werden. Durch die Auswahl von „CAx-Daten exportieren“ aus dem Reiter „Werkzeuge“ konnte die AML Datei für die Hardwarekonfiguration erzeugt werden(siehe Abb. 5.5). Dieser Schritt kann ebenfalls voll programmatisch und automatisiert per TIA Openness Api erfolgen.

Um auch das Vorhandensein aller in der Hardwarekonfiguration genannten Geräte prüfen zu können, wurde per DCP-Befehl das Netzwerk überprüft. Alle per Ethernet verbundenen Profinet Geräte wurden gefunden und die Suchdaten wurden abgespeichert.

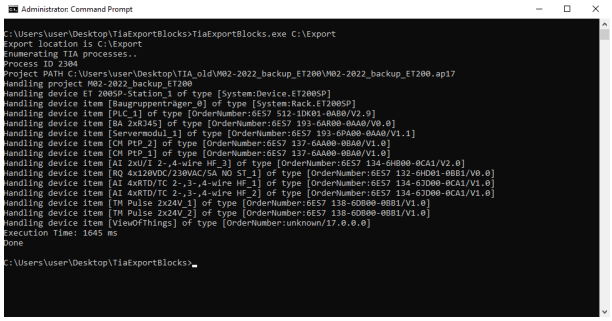


Abbildung 5.4: TiaExportBlocks nach dem Erfolgreichen Export

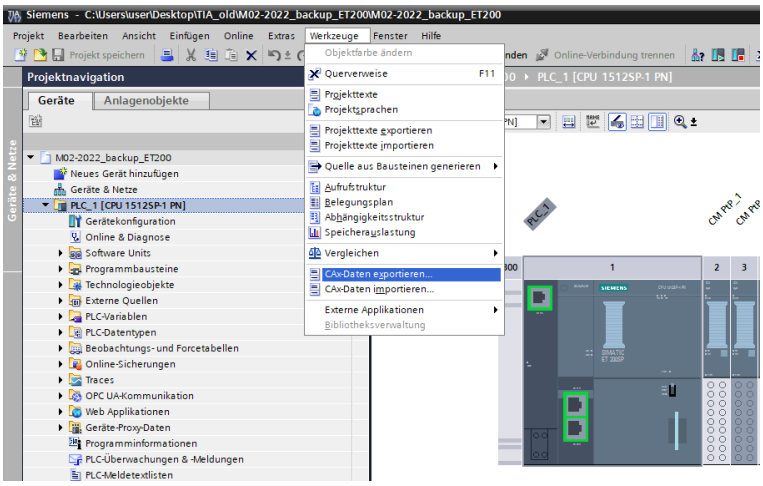


Abbildung 5.5: Hardware Ist-Zustand der SPS zu AML exportieren

Soll-Zustand ermitteln

In der dritten Phase der Kompatibilitätsprüfung wurde der Soll-Zustand des Gesamtsystems erfasst. Wie schon in Kap. 4.1.2 erwähnt gibt es dabei zwei Anwendungsfälle. In der Evaluation wurde dabei der Anwendungsfall des Aufspielens von neuer Software auf die SPS evaluiert. Dieser Schritt umfasst die Ermittlung des Soll-Zustandes der Software und der Hardware aus dem gegebenen Software-Update, welches auf die Hardware SPS übertragen werden soll.

Software Soll-Zustand ermitteln Um den Soll-Zustand zu ermitteln, wurde zunächst das Projekt mit dem bereits abgespeicherten Ist-Zustand geschlossen. Das TIA Projekt mit den Updatedaten wurde dann mithilfe

des TIA-Portals geöffnet. Ähnlich wie bei der Ermittlung des Software Ist-Zustandes wurde die Software „TiaExportBlocks“ ânschließend genutzt, um die vorhandenen Projektdaten zu XML zu exportieren.

Hardware Soll-Zustand ermitteln Nach der Software wurde auch die Hardware durch die Auswahl von „CAx-Daten exportieren“ aus dem Reiter „Werkzeuge“ als AML Datei exportiert und gesichert (siehe Abb. 5.5). Dieser Schritt ist identisch zur Ermittlung des Ist-Zustandes der Hardware.

Vergleich des Soll- und Ist-Zustand

Da zu diesem Zeitpunkt der Soll- und Ist-Zustand in XML bzw. AML vorlag, konnten durch einen Vergleich die Unterschiede zwischen den Zuständen aufgezeigt werden. Dabei fiel auf, dass ein im Software Soll-Zustand angegebenes Profinet HMI Gerät, welches durch die SPS angesprochen werden sollte, weder in der geladenen Hardwarekonfiguration der AML-Datei, noch beim DCP-Scan gefunden werden konnte.

Erkennung Inkompatibilitäten Das neue Software-Update beinhaltete eine Änderung, welche zu einer Inkompatibilität von Hardware und Software geführt hätte, sofern die Daten auf die SPS übertragen worden wären. Demnach wurden Variablen im SPS Steuerungscode verwendet, welche auf die nicht vorhandene HMI verwiesen. Als Ergebnis des Vergleiches des Soll- und Ist-Zustandes, wurden die gefundenen Inkompatibilitäten in der Fehlerdetektionstabelle aufgezeigt und für den Testingenieur sichtbar dargestellt.

Vergleich Soll und Ist-Zustand	
Erkannter Fehler	Reaktion
H/W-Gerät „HMI01“ nicht in Ist-Zustand vorhanden	Geräteverbindungen prüfen(L)

Reaktionen und Endergebnis Dabei wurde dem Testingenieur der Hinweis gegeben, die Geräteverbindungen zu prüfen. So konnte er feststellen ob es sich nur um einen Verbindungsfehler handelte. Tatsächlich wurde das angesprochene HMI Gerät am Modul gefunden. Es wurde erkannt, dass die Ethernet Verbindung des Gerätes nicht ordnungsgemäß angeschlossen war und das Gerät somit nicht gefunden werden konnte. Nachdem der Fehler

durch die adäquate Reaktion des Testingenieur lokal behoben worden war, konnte das Software-Update mithilfe des TIA-Portals auf die verbundene SPS Steuerung übertragen werden.

5.5 Erfüllung der Anforderungen an die Kompatibilitätsprüfung

Die in diesem Abschnitt durchgeführte Evaluation prüfte das Konzept auf die Erfüllung der Anforderungen. Folgende Anforderungen aus Kapitel 3.2.1 konnten dabei erfüllt werden.

Erfüllungsgrad der Anforderungen	
Anforderung	Grad der Erfüllung
3.2.1 A	Vollständig
3.2.1 B	Teilweise
3.2.1 C	Vollständig
3.2.1 D	Vollständig
3.2.1 E	Teilweise

Anforderung 3.2.1 A

Durch den Vergleich des Soll- und Ist-Zustandes können Inkompatibilitäten aufgedeckt werden und eine Aussage über die Kompatibilität zweier Komponenten getroffen werden. Diese werden dem Testingenieur zum Ende der Kompatibilitätsprüfung als Zeile in der Fehlerdetektionstabelle angezeigt. Damit konnte die Anforderung 3.2.1 A erfüllt werden.

Anforderung 3.2.1 B

Durch den Einsatz eines externen I-PC kann die Kompatibilitätsprüfung jederzeit ereignisbasiert die Kompatibilitätsprüfung durch das Durchlaufen des Testskriptes automatisiert abarbeiten. Dies funktioniert sowohl beim Neustart als auch beim Aufspielen neuer Software. Durch die Evaluation konnte diese Anforderung nur teilweise erfüllt werden, da ein Neustart der SPS nicht evaluiert worden ist.

Anforderung 3.2.1 C

Die gefundenen Fehler und Inkompatibilitäten werden in einer Fehlerdetektionstabelle angezeigt und in verschiedene Abschnitte eingeordnet. Anforderung 3.2.1 C konnte somit vollständig erfüllt werden.

Anforderung 3.2.1 D

In der Fehlerdetektionstabelle werden dem Testingenieur mögliche Reaktionen unter Angabe der Lokalität angezeigt damit dieser adäquat auf einen Fehler reagieren kann. Somit ist diese Anforderung ebenfalls vollständig erfüllt und auch evaluiert.

Anforderung 3.2.1 E

Das Konzept wurde ausschließlich für Software von Siemens entwickelt und mit Software von Siemens getestet. Wegen vorhandener Standards können die aufgezeigten Lösungen und Konzeptideen einfach auf weitere Hersteller erweitert werden. Die Weiterentwicklung des Konzeptes, um auch mit Software anderer Hersteller zu funktionieren, ist demnach noch Teil des bestehenden Forschungsbedarfs. Daher wurde diese Anforderung nur teilweise erfüllt.

6 Diskussion

Das vorgestellte Konzept der Kompatibilitätsprüfung gibt Testingenieuren von industriellen Steuerungen eine Übersicht über Inkompatibilitäten noch bevor diese während des Betriebs der Steuerung auftreten. Es werden Unterschiede zwischen dem Soll- und Ist-Zustand aufgezeigt und mögliche Reaktionen genannt.

Durch den Einsatz eines I-PC wird die Kompatibilitätsprüfung automatisch bei einem Neustart der SPS oder beim Aufspielen neuer Software durchgeführt. Somit werden Fehler und Inkonsistenzen früh aufgedeckt und können entsprechend behoben werden.

6.1 Beantwortung der Forschungsfragen

Forschungsfrage FF 1 Die Forschungsfrage FF 1 behandelte vor allem die Recherche von bestehender Literatur und deren Analyse. Durch die Beantwortung der TA 1 und TA 2 im Kapitel 2 und der Beantwortung der TA 3 und TA 4 im Kapitel 3 wurde diese erste Forschungsfrage beantwortet.

Forschungsfrage FF 2 Weiterhin wurden im Kapitel der Anforderungsanalyse (siehe Kap. 3) ebenfalls die TA 5-7 beantwortet. Dadurch konnten die wesentlichen Eigenschaften der Hardware und Softwarekomponenten festgestellt werden, um so einen Kompatibilitätstest zu ermöglichen. Damit ist der Inhalt der FF 2 ebenfalls beantwortet.

Forschungsfrage FF 3 Das in Kapitel 4 vorgestellte Konzept der Kompatibilitätsprüfung beantwortet zusammen mit dem Kapitel zur Evaluation (siehe Kap. 5) und dem aktuellen Kapitel abschließend die TA 8-12 und damit die Forschungsfrage FF 3.

6.2 Diskussion zur Forschungsmethodik

Bei der Bearbeitung der Forschungsfragen wurde sich an das Skriptum von (Biedermann et al., 2012) gehalten. Dabei wurden die vier Stufen der For-

schungsarbeit, basierend auf dem Typ zur Bearbeitung einer Lösung, erstellt. Dabei lag der Forschungsschwerpunkt auf der Erstellung des Konzeptes der Kompatibilitätsprüfung, also auf der Präskriptiven Studie. Der Typ der Studienarbeit ist somit produktgestaltend und konstruktiv und kann in die Forschungsart der angewandten Forschung eingeordnet werden.

6.3 Bewertung des Konzeptes

Das vorgestellte Konzept zur Kompatibilitätsprüfung ermöglicht einen automatisierten Vergleich des Soll- und Ist-Zustandes der Hardware und Software einer SPS. Dieser Vergleich ermöglicht die Ermittlung von Inkonsistenzen, bzw. Inkompatibilitäten zwischen den Komponenten. Dadurch kann sowohl zum Neustart der SPS als auch beim Software-Update eine Prüfung auf Modellabweichungen erfolgen. Die in Kap. 4.2 gezeigten Tabellen bieten eine Übersicht über mögliche auftretende Fehler und mögliche Reaktionen. Diese Tabellen sind nicht vollständig und umfassen eine viel größere Anzahl an möglichen Inkonsistenzen, die aufgedeckt werden können, als dargestellt. Die Tabellen wurden nicht vollständig dargestellt, um den Fokus auf das Konzept selbst nicht zu verlieren.

Das vorgestellte Konzept nutzt ausschließlich Software von Siemens und dem TIA-Portal. Für nachfolgende Arbeiten wäre eine weitere Betrachtung des Konzeptes zur Ausweitung auf weitere wichtige Hersteller von industriellen Steuerungen sinnvoll und würde zur Forschung in diesem Bereich beitragen.

Da diese Studienarbeit keine eigene Implementierung eines Testskriptes beinhaltet, wäre auch dieser Punkt eine sinnvolle Erweiterung, um die Anwendbarkeit des vorgestellten Konzeptes weiter zu überprüfen. Aufgrund der anfänglichen Eingrenzung (siehe Kap. 1.1.3) der Aufgabenstellung wurde die Implementierung des Testskriptes in dieser Arbeit nicht durchgeführt. Der zeitliche Bearbeitungsrahmen dieser Arbeit konnte somit eingehalten werden.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Diese Arbeit befasste sich mit den in Kapitel 1.1.1 genannten Forschungsfragen und Teilaufgaben. Dabei wurde durch Literaturrecherche, Angewandte Forschung und Anforderungsanalysen eine Teststrategie zur Kompatibilitätsprüfung von Hardware und Software in industriellen Steuerungen entwickelt.

Im Kapitel zum Stand der Technik wurde auf vorhandene Teststrategien aus der VIBN und der CS eingegangenen und der Begriff von Kompatibilität aus verschiedenen Literaturquellen zusammengeführt und beleuchtet. Die Strategien SiL und HiL aus der VIBN und die anomaliebasierte Erkennung aus der CS wurden dabei als besonders relevant und maßgebend für die in dieser Studienarbeit vorgestellten Kompatibilitätsprüfung angesehen.

Darauf basierend wurden dann mittels Anforderungsanalyse die wichtigsten Anforderungen an das Konzept hergeleitet und aufgezeigt. Weiterhin wurde darauf eingegangen, wie die ermittelten Strategien aus der VIBN und der CS auf die Prozesse des Neustarts der SPS und dem Aufspielen neuer Software auf die SPS übertragen werden können.

Das vorgestellte Konzept zur Durchführung der Kompatibilitätsprüfung wurde mittels angewandter Forschung in Bezug auf die frühzeitige Feststellung von Inkompatibilitäten erstellt. Es wurde ein 4-Phasen Konzept vorgestellt, welches sich durch den Vergleich von Soll- und Ist-Zuständen der Software- und Hardwarekomponenten auszeichnet.

Als Ergebnis der Kompatibilitätsprüfung offeriert das Konzept eine Übersicht über gefundene Inkompatibilitäten und zeigt mögliche Reaktionen auf.

Die Funktionsweise des Konzeptes wurde an einem Modul des P2O-Labs der TU Dresden durchgeführt und evaluiert. Die Ergebnisse entsprachen dabei den Konzeptanforderungen und ermöglichten eine Erkennung von Inkompatibilitäten.

In der an die Evaluation anschließende Diskussion wurde das vorgestellte Konzept abschließend bewertet und eine Reflexion zur Forschungsmethodik dargelegt.

7.2 Ausblick und weitere Forschungsarbeit

Für weitere Forschungsarbeit rund um den Bereich der Kompatibilitätsprüfung bieten sich zahlreiche Möglichkeiten. Diese Möglichkeiten können diese Arbeit als theoretische Basis nehmen und verschiedenste Implementierungen des vorgestellten Konzeptes durchführen. Das Konzept kann auch für weitere Software und Hardwarekonfigurationen von verschiedenen anderen Herstellern evaluiert und erweitert werden.

Weiterhin ist die Programmierung eines Testskriptes, welches die verschiedenen Phasen des vorgestellten Konzeptes automatisiert abarbeitet, eine wichtige Erweiterung zu dieser Studienarbeit. Eine weitaus größere Erweiterung des Konzeptes, um beispielsweise Daten aus weiteren Quellen zur Ermittlung des Soll- und Ist-Zustandes zu beziehen, ist ebenfalls eine Möglichkeit, um die Robustheit und Anwendbarkeit des vorgestellten Konzeptes zusätzlich zu festigen.

In Zukunft könnten vollautomatisierte Kompatibilitätstestsysteme den Alltag von Testingenieuren und Integratoren von industriellen Steuerungen stark erleichtern. Somit könnten Inkompatibilitäten in verschiedensten Arten von Industrie 4.0 Komponenten frühzeitig gefunden werden. Gerade bei immer mehr Neuentwicklungen von Hardware- und Softwarekomponenten sowie Veränderungen bestehender SPS-Architekturen können automatisierte Kompatibilitätstests eine große Hilfe sein.

Literaturverzeichnis

acatech (2013). *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0. Abschlussbericht des Arbeitskreises Industrie 4.0*. de-DE. URL: https://www.acatech.de/publikation/umsetzungsempfehlungen-fuer-das-zukunftsprojekt-industrie-4-0-abschlussbericht-des-arbeitskreises-industrie-4-0/ (besucht am 17. 06. 2022).

AutomationML (2022). *What is AutomationML? – AutomationML*. de-DE. URL: https://www.automationml.org/about-automationml/automationml/ (besucht am 03. 08. 2022).

Bartz, Patrick, Michael Decottignies, Yann Montagnol, Khalid Kouiss, Fabio Pedrotti Terra und Cláudio Luís D'Elia Machado (Dez. 2021). „Manufacturing machine virtual commissioning: Automated validation of the control software". In: *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, S. 1–5. DOI: 10.1109/ICECET52533.2021.9698716.

Berre, Arne-jørgen, Axel Hahn, David Akehurst, Aphrodite Tsalgatidou, François Vermaut, Peter F. Linington und Workflows (2004). *Initial Draft – part V – Agent*.

Biedermann, Wieland, Katharina Kirner, Maximilian Kissel, Stefan Langer, Christopher Münzberg und Martina Wickel (2012). „Forschungsmethodik in den Ingenieuerwissenschaften". de. In: S. 70.

BSI (2014). „Die Lage der IT-Sicherheit in Deutschland 2014". de. In: S. 44.

Chen, David, Guy Doumeingts und François Vernadat (Sep. 2008). „Architectures for enterprise integration and interoperability: Past, present and future". en. In: *Computers in Industry*. Enterprise Integration and Interoperability in Manufacturing Systems 59.7, S. 647–659. ISSN: 0166-3615. DOI: 10.1016/j.compind.2007.12.016. URL: https://www.sciencedirect.com/science/article/pii/S0166361508000365 (besucht am 23. 06. 2022).

Chen, David und François Vernadat (Jan. 2002). „Enterprise Interoperability: A Standardisation View“. In: Bd. 108, S. 273–282. ISBN: 978-1-4757-5151-2. DOI: 10.1007/978-0-387-35621-1_28.

Chituc, Claudia-Melania, César Toscano und Americo Azevedo (Sep. 2008). „Interoperability in Collaborative Networks: Independent and industry-specific initiatives – The case of the footwear industry“. en. In: *Computers in Industry*. Enterprise Integration and Interoperability in Manufacturing Systems 59.7, S. 741–757. ISSN: 0166-3615. DOI: 10.1016/j.compind.2007.12.012. URL: https://www.sciencedirect.com/science/article/pii/S016636150800047X (besucht am 23. 06. 2022).

Collins, Eliane Figueiredo und Vicente Ferreira de Lucena (Juni 2012). „Software test automation practices in agile development environment: an industry experience report“. In: *Proceedings of the 7th International Workshop on Automation of Software Test*. AST '12. Zurich, Switzerland: IEEE Press, S. 57–63. ISBN: 978-1-4673-1822-8. (Besucht am 18. 06. 2022).

Decker, S., S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann und I. Horrocks (Sep. 2000). „The Semantic Web: the roles of XML and RDF“. In: *IEEE Internet Computing* 4.5. Conference Name: IEEE Internet Computing, S. 63–73. ISSN: 1941-0131. DOI: 10.1109/4236.877487.

Fielding, Roy Thomas (2000). *Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)*. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (besucht am 23. 06. 2022).

Ford, Thomas (1980). „A Survey on Interoperability Measurement“. en. In: S. 28.

Gaddam, S.R., V.V. Phoha und K.S. Balagani (2007). „K-Means+ID3: A novel method for supervised anomaly detection by cascading k-Means clustering and ID3 decision tree learning methods“. English. In: *IEEE Transactions on Knowledge and Data Engineering* 19.3, S. 345–354. ISSN: 1041-4347. DOI: 10.1109/TKDE.2007.44.

Gräßler, Iris, Tobias Bruckmann, Michael Dattner, Thomas Ehl, Martin Hawlas, Julian Hentze, Philipp Hesse, Christoph Termühlen, Roland Lachmayer, Marvin Knöchelmann, Randolf Mock, I. Mozgova, Daniel Preuß,

Maximilian Schneider, Guido Stollt, Henrik Thiele und Dominik Wiechel (Nov. 2021). *VDI/VDE 2206: Entwicklung mechatronischer und cyber-physischer Systeme - Inhaltsverzeichnis*.

Heiler, Sandra (Juni 1995). „Semantic interoperability“. In: *ACM Computing Surveys* 27.2, S. 271–273. ISSN: 0360-0300. DOI: 10.1145/210376.210392. URL: https://doi.org/10.1145/210376.210392 (besucht am 23. 06. 2022).

Helsinki, University of, F. Eliassen und J. Veijalainen, Hrsg. (o.D.). *A functional approach to information system interoperability*.

Hill, Rafael Balderas, Justine Delbos, Selena Trebosc, Junior Tsague, Gregoire Feroldi, Jessy Martin, Tobiah Master und Nicolas Lassabe (Sep. 2021). „Improving interoperability of Virtual Commissioning toolchains by using OPC-UA-based technologies“. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, S. 01–07. DOI: 10.1109/ETFA45728.2021.9613490.

Ho, Yuan, Deborah Frincke und Donald Jr. Tobin (1998). *21st National Information Systems Security Conference: Building the Information Security Bridge to the 21st Century : October 5-8, 1998, Hyatt Regency Crystal City, Arlington, Va*. en. Google-Books-ID: 5g0SAQAAMAAJ. National Institute of Standards und Technology.

Hoffmann, Peter, Talal M.A. Maksoud, Reimar Schumann und Giuliano C. Premier (Juni 2010). „Virtual Commissioning Of Manufacturing Systems A Review And New Approaches For Simplification“. en. In: *ECMS 2010 Proceedings edited by A Bargiela S A Ali D Crowley E J H Kerckhoffs*. ECMS, S. 175–181. ISBN: 978-0-9564944-0-5. DOI: 10.7148/2010-0175-0181. URL: http://www.scs-europe.net/dlib/2010/2010-0175.htm (besucht am 12. 06. 2022).

Hofman, Wout (2019). „Toward Large-Scale Logistics Interoperability Based on an Analysis of Available Open Standards“. en. In: *Enterprise Interoperability VIII*. Hrsg. von Keith Popplewell, Klaus-Dieter Thoben, Thomas Knothe und Raúl Poler. Proceedings of the I-ESA Conferences. Cham: Springer International Publishing, S. 249–261. ISBN: 978-3-030-13693-2. DOI: 10.1007/978-3-030-13693-2_21.

IEC 62424 (2016). *IEC 62424 - Ed. 1.0. Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools.* Techn. Ber.

IEEE Std 610.12-1990 (Dez. 1990). „IEEE Standard Glossary of Software Engineering Terminology“. In: *IEEE Std 610.12-1990*. Conference Name: IEEE Std 610.12-1990, S. 1–84. DOI: 10.1109/IEEESTD.1990.101064.

Ilgun, K., R.A. Kemmerer und P.A. Porras (1995). „State Transition Analysis: A Rule-Based Intrusion Detection Approach“. English. In: *IEEE Transactions on Software Engineering* 21.3, S. 181–199. ISSN: 0098-5589. DOI: 10.1109/32.372146.

ISO 16100-1:2009 (2009). *ISO 16100-1:2009 Industrial automation systems and integration — Manufacturing software capability profiling for interoperability — Part 1: Framework*. en. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/05/33/53378.html (besucht am 23. 06. 2022).

ISO/IEC 19941:2017 (2017). *ISO/IEC 19941:2017 Information technology — Cloud computing — Interoperability and portability*. en. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/66/66639.html (besucht am 23. 06. 2022).

ISO/IEC 20922:2016 (2016). *ISO/IEC 20922:2016 Information technology — Message Queuing Telemetry Transport (MQTT) v3.1.1*. en. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/94/69466.html (besucht am 23. 06. 2022).

ISO/IEC 2382:2015 (2015). *ISO/IEC 2382:2015 Information technology — Vocabulary*. en. URL: https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/35/63598.html (besucht am 21. 06. 2022).

Kasunic, Mark (2001). *Measuring Systems Interoperability*.

Kaur, Parneet, Manish Kumar und Abhinav Bhandari (Jan. 2017). „A review of detection approaches for distributed denial of service attacks“. In: *Systems Science & Control Engineering* 5.1. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/21642583.2017.1331768, S. 301–320. ISSN:

null. DOI: 10.1080/21642583.2017.1331768. URL: https://doi.org/10.1080/21642583.2017.1331768 (besucht am 19. 06. 2022).

Kemmerer, Sharon (Feb. 2009). *Manufacturing Interoperability Program, a Synopsis*. DOI: 10.13140/RG.2.1.2567.9440.

Kiljander, Jussi, Alfredo D'elia, Francesco Morandi, Pasi Hyttinen, Janne Takalo-Mattila, Arto Ylisaukko-Oja, Juha-Pekka Soininen und Tullio Salmon Cinotti (2014). „Semantic interoperability architecture for pervasive computing and internet of things". In: *IEEE access* 2. Publisher: IEEE, S. 856–873.

Koren, Y., U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy und H. Van Brussel (Jan. 1999). „Reconfigurable Manufacturing Systems". en. In: *CIRP Annals* 48.2, S. 527–540. ISSN: 0007-8506. DOI: 10.1016/S0007-8506(07)63232-6. URL: https://www.sciencedirect.com/science/article/pii/S0007850607632326 (besucht am 17. 06. 2022).

Kumar, Sandeep und Eugene H Spafford (1994). „A Pattern Matching Model for Misuse Intrusion Detection". en. In: S. 15.

Langner, R. (2011). „Stuxnet: Dissecting a cyberwarfare weapon". English. In: *IEEE Security and Privacy* 9.3, S. 49–51. ISSN: 1540-7993. DOI: 10.1109/MSP.2011.67.

Lechler, Tobias, Eva Fischer, Maximilian Metzner, Andreas Mayr und Jörg Franke (Jan. 2019). „Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems". en. In: *Procedia CIRP*. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019 81, S. 1125–1130. ISSN: 2212-8271. DOI: 10.1016/j.procir.2019.03.278. URL: https://www.sciencedirect.com/science/article/pii/S2212827119305839 (besucht am 12. 06. 2022).

Lee, Chi G. und Sang C. Park (Juli 2014). „Survey on the virtual commissioning of manufacturing systems". en. In: *Journal of Computational Design and Engineering* 1.3, S. 213–222. ISSN: 2288-4300. DOI: 10.7315/JCDE.2014.021. URL: https://www.sciencedirect.com/science/article/pii/S2288430014500292 (besucht am 12. 06. 2022).

Lee, Keunsoo, Juhyun Kim, Ki Hoon Kwon, Younggoo Han und Sehun Kim (Apr. 2008). „DDoS attack detection method using cluster analysis". en.

In: *Expert Systems with Applications* 34.3, S. 1659–1665. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2007.01.040. URL: https://www.sciencedirect.com/science/article/pii/S0957417407000395 (besucht am 19. 06. 2022).

Lelli, Francesco (Feb. 2019). „Interoperability of the Time of Industry 4.0 and the Internet of Things“. In: *Future Internet* 11, S. 36. DOI: 10.3390/fi11020036.

Lezzi, Marianna, Mariangela Lazoi und Angelo Corallo (Dez. 2018). „Cybersecurity for Industry 4.0 in the current literature: A reference framework“. en. In: *Computers in Industry* 103, S. 97–110. ISSN: 01663615. DOI: 10.1016/j.compind.2018.09.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166361518303658 (besucht am 21. 06. 2022).

Liao, Yihua und V. Rao Vemuri (Okt. 2002). „Use of K-Nearest Neighbor classifier for intrusion detection11An earlier version of this paper is to appear in the Proceedings of the 11th USENIX Security Symposium, San Francisco, CA, August 2002“. en. In: *Computers & Security* 21.5, S. 439–448. ISSN: 0167-4048. DOI: 10.1016/S0167-4048(02)00514-X. URL: https://www.sciencedirect.com/science/article/pii/S0167404802005140X (besucht am 19. 06. 2022).

Luder, A., A. Klostermeyer, J. Peschke, A. Bratoukhine und T. Sauter (Feb. 2005). „Distributed automation: PABADIS versus HMS“. In: *IEEE Transactions on Industrial Informatics* 1.1. Conference Name: IEEE Transactions on Industrial Informatics, S. 31–38. ISSN: 1941-0050. DOI: 10.1109/TII.2005.843825.

Machado, Jose und Eurico Seabra (Juli 2013). „HiL simulation workbench for testing and validating PLC programs“. In: *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. ISSN: 2378-363X, S. 230–235. DOI: 10.1109/INDIN.2013.6622887.

Magnus, Stephan, Tim Russ und Jan Krause (Mai 2016). „Anforderungs- und modellbasierte Testfallgenerierung und Testdurchführung unter Nutzung von Methoden zur Netzwerkanalyse“. In:

Maurushat, Alana (2013). *Disclosure of Security Vulnerabilities*. en. SpringerBriefs in Cybersecurity. London: Springer London. ISBN: 978-1-4471-5003-

9. DOI: 10.1007/978-1-4471-5004-6. URL: http://link.springer.com/10.1007/978-1-4471-5004-6 (besucht am 19. 06. 2022).

Mazza, Luigi (2018). „Virtual commissioning of a pneumatic servosystem with a PLC in MiL, SiL, and HiL“. en. In: S. 99.

Mcfarlane, Duncan C. und Stefan Bussmann (Jan. 2000). „Developments in holonic production planning and control“. en. In: *Production Planning & Control* 11.6, S. 522–536. ISSN: 0953-7287, 1366-5871. DOI: 10.1080/095372800414089. URL: https://www.tandfonline.com/doi/full/10.1080/095372800414089 (besucht am 08. 06. 2022).

Morris, Edwin und Linda Levine (2004). „System of Systems Interoperability (SOSI): Final Report“. en. In: S. 67.

Mosley, Mark, Michael H. Brackett, Susan Earley, Deborah Henderson und Data Administration Management Association (2010). *The DAMA guide to the data management body of knowledge: (DAMA-DMBOK Guide)*. eng. 1st ed. Bradley Beach, New Jersey: Technics Publications. ISBN: 978-1-935504-02-3.

Mukkamala, S., G. Janoski und A. Sung (2002). „Intrusion detection using neural networks and support vector machines“. English. In: Bd. 2, S. 1702–1707.

Muresan, Marius und Dan Pitica (Okt. 2012). „Software in the Loop environment reliability for testing embedded code“. In: *2012 IEEE 18th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, S. 325–328. DOI: 10.1109/SIITME.2012.6384402.

NBC News (2014). *Cybercrime Costs Businesses $445 Billion and Thousands of Jobs: Study*. en. URL: https://www.nbcnews.com/tech/security/cybercrime-costs-businesses-445-billion-thousands-jobs-study-n124746 (besucht am 19. 06. 2022).

Oppelt, Mathias, Gerrit Wolf und Leon Urbas (Sep. 2015). „Towards an integrated use of simulation within the life-cycle of a process plant“. In: *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. ISSN: 1946-0759, S. 1–8. DOI: 10.1109/ETFA.2015.7301521.

Ourston, D., S. Matzner, W. Stump und B. Hopkins (2003). „Applications of hidden Markov models to detecting multi-stage network attacks“. English. In: S. 10–19. ISBN: 978-0-7695-1874-9. DOI: 10.1109/HICSS.2003.1174909.

Ovatman, Tolga, Atakan Aral, Davut Polat und Ali Ünver (Dez. 2014). „An overview of model checking practices on verification of PLC software“. In: *Software & Systems Modeling* 15, S. 1–24. DOI: 10.1007/s10270-014-0448-7.

Pan, Z.-S., S.-C. Chen, G.-B. Hu und D.-Q. Zhang (2003). „Hybrid neural network and C4.5 for misuse detection“. English. In: Bd. 4, S. 2463–2467. ISBN: 978-0-7803-7865-0.

Pantsar-Syväniemi, Susanna, Anu Purhonen, Eila Ovaska, Jarkko Kuusijärvi und Antti Evesti (2012). „Situation-based and self-adaptive applications for the smart environment“. In: *Journal of Ambient Intelligence and Smart Environments* 4.6. Publisher: IOS Press, S. 491–516.

PLCopen foundation (2009). „XML Formats for IEC 61131-3“. en. In: S. 80.

Plummer, A. R. (Mai 2006). „Model-in-the-Loop Testing“. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 220.3. Publisher: IMECHE, S. 183–199. ISSN: 0959-6518. DOI: 10.1243/09596518JSCE207. URL: https://doi.org/10.1243/09596518JSCE207 (besucht am 17. 06. 2022).

PROFINET University (Juni 2018). *DCP - Discovery and Configuration Protocol*. en-US. URL: https://profinetuniversity.com/naming-addressing/profinet-dcp/ (besucht am 24. 08. 2022).

Radatz, J, A Geraci und F Katki (1990). „IEEE standard glossary of software engineering terminology. IEEE Std. 610.12-1990“. In: *Computer Society of the IEEE*.

Rafi, Dudekula Mohammad, Katam Reddy Kiran Moses, Kai Petersen und Mika V. Mäntylä (Juni 2012). „Benefits and limitations of automated software testing: systematic literature review and practitioner survey“. In: *Proceedings of the 7th International Workshop on Automation of Software Test*. AST '12. Zurich, Switzerland: IEEE Press, S. 36–42. ISBN: 978-1-4673-1822-8. (Besucht am 18. 06. 2022).

Ray, S. R. und A. T. Jones (Dez. 2006). „Manufacturing interoperability“. en. In: *Journal of Intelligent Manufacturing* 17.6, S. 681–688. ISSN: 1572-8145. DOI: 10.1007/s10845-006-0037-x. URL: https://doi.org/10.1007/s10845-006-0037-x (besucht am 23. 06. 2022).

Raymond, Brian (2016). „Cybersecurity in the Manufacturing Sector“. en. In: S. 5.

Raymond, Dr und J. Curts (1999). *Architecture: The Road to Interoperability*.

Sabhnani, M. und G. Serpen (2003). „Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context“. English. In: S. 209–215. ISBN: 978-1-932415-11-7.

Shukla, Aadya, Steve Harris und Jim Davies (2010). „Semantic Interoperability in Practice“. en. In: *2010 43rd Hawaii International Conference on System Sciences*. Honolulu, Hawaii, USA: IEEE, S. 1–10. ISBN: 978-1-4244-5509-6. DOI: 10.1109/HICSS.2010.334. URL: http://ieeexplore.ieee.org/document/5428305/ (besucht am 23. 06. 2022).

Siemens (Aug. 2018). *Wie können Sie in STEP 7 (TIA Portal) die "Programmbausteine" mit den PLC-Variabl... - ID: 41885693 - Industry Support Siemens*. URL: https://support.industry.siemens.com/cs/document/41885693/wie-k%C3%B6nnen-sie-in-step-7-(tia-portal)-die-programmbausteine-mit-den-plc-variablen-und-den-plc-datentypen-von-der-cpu-in-das-programmierger%C3%A4t-laden-?dti=0&lc=de-AT (besucht am 06. 09. 2022).

Siemens Openness Api Handbuch (Mai 2021). „Openness: API für die Automatisierung von Engineering-Workflows“. de. In: S. 1316.

Siemens SAT Handbuch (Sep. 2021). „SIMATIC Automation Tool V4.0 SP3 Benutzerhandbuch“. de. In: S. 376. URL: https://support.industry.siemens.com/cs/document/109801801/simatic-automation-tool-v4-0-sp3-benutzerhandbuch?dti=0&lc=de-DE (besucht am 24. 08. 2022).

Suteu, Cezar (Apr. 2020). *TiaExportBlocks: Tia Openness Export Blocks*. URL: https://github.com/Speedstuff/TiaExportBlocks (besucht am 05. 09. 2022).

Thames, L. und D. Schaefer (2017). *Cybersecurity for Industry 4.0*. en. URL: https://link.springer.com/book/10.1007/978-3-319-50660-9 (besucht am 19. 06. 2022).

Thramboulidis, Kleanthis (Jan. 2010). „The 3+1 SysML View-Model in Model Integrated Mechatronics“. In: *JSEA* 3, S. 109–118. DOI: 10.4236/jsea.2010.32014.

Tolk, A. (2004). „Composable Mission Spaces and M&S Repositories - Applicability of Open Standards“. In:

Ugarte, Miriam, Leire Etxeberria, Gorka Unamuno, Jose Luis Bellanco und Eneko Ugalde (2022). „Implementation of Digital Twin-based Virtual Commissioning in Machine Tool Manufacturing“. en. In: *Procedia Computer Science* 200, S. 527–536. ISSN: 18770509. DOI: 10.1016/j.procs.2022.01.250. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050922002599 (besucht am 12. 06. 2022).

Ulewicz, Sebastian, Daniel Schütz und Birgit Vogel-Heuser (Okt. 2014). „Software changes in factory automation: Towards automatic change based regression testing“. In: *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*. ISSN: 1553-572X, S. 2617–2623. DOI: 10.1109/IECON.2014.7048875.

Veer, H. van der und A. Wiles (2008). „Achieving Technical Interoperability-the ETSI Approach“. In: *European Telecommunications Standards Institute: Sophia Antipolis*.

Vigna, G. und R.A. Kemmerer (Dez. 1998). „NetSTAT: a network-based intrusion detection approach“. In: *Proceedings 14th Annual Computer Security Applications Conference (Cat. No.98EX217)*. ISSN: 1063-9527, S. 25–34. DOI: 10.1109/CSAC.1998.738566.

Warsinske, John (2019). *CISSP: Certified Information Systems Security Professional*. 1. Aufl. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119423300. John Wiley & Sons, Ltd. DOI: 10.1002/9781119423300. URL: https://onlinelibrary.wiley.com/doi/10.1002/9781119423300 (besucht am 19. 06. 2022).

Wu, Dazhong, Anqi Ren, Wenhui Zhang, Feifei Fan, Peng Liu, Xinwen Fu und Janis Terpenny (Juli 2018). „Cybersecurity for digital manufacturing“. en. In: *Journal of Manufacturing Systems* 48, S. 3–12. ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.03.006. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612518300396 (besucht am 19. 06. 2022).

Xu, Xun (Feb. 2012). „From cloud computing to cloud manufacturing“. en. In: *Robotics and Computer-Integrated Manufacturing* 28.1, S. 75–86. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2011.07.002. URL: https://www.sciencedirect.com/science/article/pii/S0736584511000949 (besucht am 23. 06. 2022).

Zäh, Michael F., Georg Wünsch, Thomas Hensel und Alexander Lindworsky (Okt. 2006). „Nutzen der virtuellen Inbetriebnahme: Ein Experiment“. de. In: *Zeitschrift für wirtschaftlichen Fabrikbetrieb* 101.10. Publisher: De Gruyter, S. 595–599. ISSN: 2511-0896. DOI: 10.3139/104.101070. URL: https://www.degruyter.com/document/doi/10.3139/104.101070/html (besucht am 18. 06. 2022).

Zeid, Abe, Sarvesh Sundaram, Mohsen Moghaddam, Sagar Kamarthi und Tucker Marion (Apr. 2019). „Interoperability in Smart Manufacturing: Research Challenges“. en. In: *Machines* 7.2, S. 21. ISSN: 2075-1702. DOI: 10.3390/machines7020021. URL: https://www.mdpi.com/2075-1702/7/2/21 (besucht am 23. 06. 2022).

Zeller, Andreas (2019). *9783844070392 - Absicherung von verteilten Automatisierungssystemen nach Änderungen der Steuerungssoftware - Andreas Zeller*. de-de. URL: https://www.eurobuch.com/buch/isbn/9783844070392.html (besucht am 30. 05. 2022).

Zeller, Andreas und Michael Weyrich (Sep. 2016). „Challenges for functional testing of reconfigurable production systems“. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, S. 1–4. DOI: 10.1109/ETFA.2016.7733620.

Zhang, Feng (2011). „Automated testing tool for PLC based industrial applications“. en. Diss. UC San Diego. URL: https://escholarship.org/uc/item/0bw1m6mp (besucht am 26. 05. 2022).

Zhang, Junzhi, Guidong Liu, Wensheng Yu und Minggao Ouyang (Mai 2008). „Adaptive control of the airflow of a PEM fuel cell system“. en. In: *Journal*

of Power Sources 179.2, S. 649–659. ISSN: 0378-7753. DOI: 10.1016/j.jpowsour.2008.01.015. URL: https://www.sciencedirect.com/science/article/pii/S0378775308001067 (besucht am 19. 06. 2022).

Selbstständigkeitserklärung

Hiermit versichere ich, Marcus Rothhaupt, geboren am 27.08.1998 in Fulda, dass ich die vorliegende Studienarbeit zum Thema

Teststrategien für Software- und Hardwarekompatibilität in industriellen Steuerungen

ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten:

Dipl.-Ing. Lucas Vogt, Dipl.-Ing. Anselm Klose

Weitere Personen waren an der geistigen Herstellung der vorliegenden Studienarbeit nicht beteiligt. Mir ist bekannt, dass die Nichteinhaltung dieser Erklärung zum nachträglichen Entzug des Diplomabschlusses (Masterabschlusses) führen kann.

Dresden, den 19.09.2022