18. AALE-Konferenz
Pforzheim, 09.03-11.03.2022

Konzept zur Automatisierung der Konfiguration einer virtuellen Inbetriebnahme

T. Neuner[1], M. Brela[2]

Zusammenfassung

Laut VDMA gewinnt die virtuelle Inbetriebnahme als umsetzungsnaher Teil der Digitalisierung im Maschinen- und Anlagenbau in allen Branchenzweigen an Bedeutung. Um die Fehlerfreiheit eines Steuerungsprogramms gewährleisten zu können, gibt es aktuell verschiedene Ansätze zur automatisierten virtuellen Inbetriebnahme. Hierbei werden definierte Testszenarien von einer Test-Applikation autonom, ohne ein Eingreifen des Testpersonals, durchgeführt. Die Herausforderung der automatisierten virtuellen Inbetriebnahme besteht in der manuellen Konfiguration der Testszenarien, die zeit- und kostenintensiv ist. Darüber hinaus ergeben sich durch die manuelle Implementierung zusätzliche Fehlerpotentiale. Diese Arbeit untersucht, wie die Konfiguration der Testszenarien automatisiert werden kann. Dabei sollen die Testszenarien direkt aus dem Steuerungsprogramm oder weiteren Engineering-Werkzeugen, wie Elektrokonstruktionen oder CAD-Pläne, abgeleitet werden. Außerdem werden Normen, wie beispielsweise die EG-Maschinenrichtlinie, berücksichtigt.

Stichwörter

Virtuelle Inbetriebnahme, Testautomatisierung, Konfigurator

1 Virtuelle Inbetriebnahme

ISG-Stuttgart beschreibt die virtuelle Inbetriebnahme als eine Simulationsmethode, mit welcher alle Anlagenfunktionalitäten anhand eines Simulationsmodells, oder unter Verwendung eines digitalen Zwillings, virtuell getestet werden können. Dabei werden Prozesszustände durch die Simulation generiert und die Reaktion des Steuerungsprogramms auf die vordefinierten Testszenarien geprüft. Durch die Simulationsmodelle wird ermöglicht, verschiedene Steuerfunktionalitäten bereits parallel zur Engineering-Phase zu testen, ohne dass eine Hardware bereits vorhanden ist. Somit können Planungs- und Softwarefehler sowie Optimierungspotentiale frühzeitig im Projektverlauf erkannt werden [1]. Je später solche Fehler im Projektverlauf lokalisiert werden, desto größer wird der Aufwand bei der Fehlerbeseitigung. Mit Hilfe der virtuellen Inbetriebnahme soll somit insgesamt der Aufwand der Inbetriebnahme an der realen Anlage reduziert werden [2].
Bei der automatisierten virtuellen Inbetriebnahme werden die Testszenarien manuell programmiert oder modelliert und anschließend automatisiert abgearbeitet. Des Weiteren wird zur Dokumentation ein Testprotokoll angelegt [3], [4], [5]. Allerdings ist der manuelle Entwurfsprozess zeitaufwändig und fehleranfällig [6].

1 B. Eng. T. Neuner
2 Prof. Dr. M. Brela

2 Handlungsbedarf

Derzeit finden sich keine Konzepte oder Werkzeuge, den Konfigurationsprozess der Testszenarien zu automatisieren. Zu diesen Testszenarien gehören u. a. Softwaretests, wie z. B. Kommunikationstests, Tests der Meldefunktionen, Überprüfung des Fehlerhandlings, Tests der Sicherheitsfunktionen oder der Prozessabläufe. Im Rahmen dieser Arbeit wurden softwaretechnische Möglichkeiten einer intelligenten Generierung der Testszenarien aus projektspezifischen Dokumenten, wie beispielsweise Steuerungsprogrammen, CAD-Zeichnungen oder Elektropläne, untersucht.

Zur Konfiguration der virtuellen Inbetriebnahme wird auf eine Bibliothek mit vordefinierten Testszenarien und Simulationsbausteinen zurückgegriffen, welche auf Basis von Projektinformationen eigenständig ausgewählt und zu einem Testprogramm zusammengesetzt werden. Dabei werden weitere Testanforderungen, wie beispielsweise die Dokumentation der Testergebnisse, integriert. Die Einbindung eines Konfigurators in den Prozess zur virtuellen Inbetriebnahme ist in folgender Grafik dargestellt.

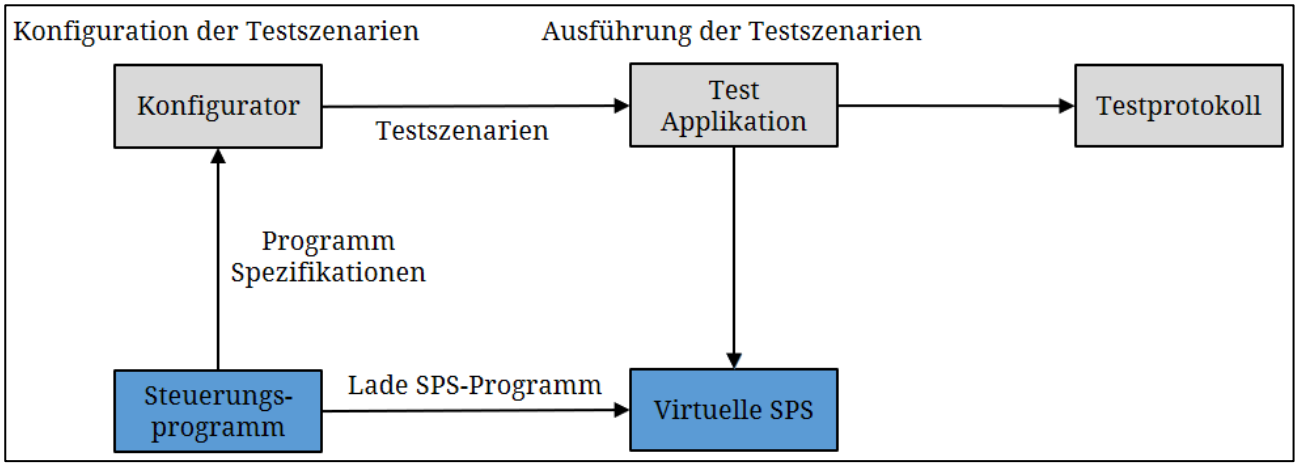

Bild 1: Einbindung eines Konfigurators in den Prozess der virtuellen Inbetriebnahme

Ziel der Studie war es, Konzepte zu entwickeln, um u. a. aus dem Steuerungsprogramm automatisiert Informationen zur Generierung einer Test-Applikation auszulesen. Dabei war die Herausforderung, dass sich sowohl die Semantik, die Syntax als auch die Programmstruktur der Steuerungsprogramme untereinander stark unterscheiden können. Dies ist darauf zurückzuführen, dass viele Unternehmen unterschiedliche interne Standards bzw. Programmierrichtlinien verwenden und es unterschiedliche Hersteller für Steuerungs- und Engineering-Systeme am Markt gibt, welche wiederum Normen, wie die DIN EN 61131, individuell umsetzen.

Durch die Anwendung eines Konfigurators zur virtuellen Inbetriebnahme können die Vorteile einer Testautomatisierung auf die automatisierte virtuelle Inbetriebnahme übertragen werden. Steirer beschreibt u. a. folgende Vorteile der Testautomatisierung im Buch Basiswissen Testautomatisierung:

- Reduktion der Programmierdauer der Testszenarien
- Reduktion des Fehlerpotentials der Testszenarien
- Wiederverwendbarkeit bereits generierter Testszenarien
- Steigerung der Testeffizienz (schnellere virtuelle Inbetriebnahme)
- Reduktion projektspezifischer Kosten
- Automatisierte Testdokumentation

3 Konfigurator zur automatisierten virtuellen Inbetriebnahme

Prinzipiell muss zur Konfiguration von Testszenarien die Programmstruktur automatisiert erfasst werden. Außerdem müssen Variablendeklarationen, Schnittstellen, Aufrufsystematiken, usw. ausgelesen und in eine zweite Projektstruktur überführt werden, welche anschließend als Vorlage für eine Test-Applikation dient. Durch den Konfigurator wird die Test-Applikation anschließend mit anwendungsspezifischen, vordefinierten Testszenarien angereichert. Hierzu wurden zwei unterschiedliche Konzepte untersucht.

3.1 Funktionsanalyse durch PLCopenXML

Zum Datenaustausch zwischen Steuerungsprogramm und Konfigurator wurde der Standard PLCopenXML ausgewählt, welcher auf dem textbasierten XML-Format beruht. Der Grund für diese Entscheidung war, dass PLCopenXML die Möglichkeit bietet, Daten eines Steuerungsprogramms zwischen unterschiedlichen Programmierumgebungen einfach auszutauschen [8]. Das generierte PLCopenXML-Dokument enthält allerdings Informationen, die zur Konfiguration der Testszenarien nicht benötigt werden. Neben dem Programmcode werden beispielsweise verwendete Bibliotheken und Visualisierungsinformationen im XML-Dokument abgelegt. Dies hat ein umfangreiches Dokument mit einer großen Anzahl an Knoten zur Folge, was anschließend die Navigation durch das Dokument zur Extraktion der Programmfunktionalitäten erschwert. Weiterhin müssen zur Navigation durch das XML-Dokument alle Knoten, in welchen die Programminformationen gespeichert werden, vorab bekannt sein. Daher wurde im Rahmen dieser Arbeit ein weiteres Konzept untersucht, um eine Programmanalyse auch ohne Vorkenntnisse der Programmstruktur durchführen zu können.

3.2 Analyse der Speicherablage des Steuerungsprogramms

Zum Datenaustausch zwischen Steuerungsprogramm und Konfigurator wurde in diesem Konzept das gezielte Zugreifen auf Dateien in der Speicherablage der Projektmappe des Steuerungsprogramms untersucht. Der Aufbau der Projektmappen basiert häufig auf einer Ordnerstruktur, welche u. a. Bibliotheken, Referenzen, Programmbausteine und Visualisierungselemente enthält. Dieser Aufbau findet sich bei den führenden Herstellern für Steuerungstechnik wieder.

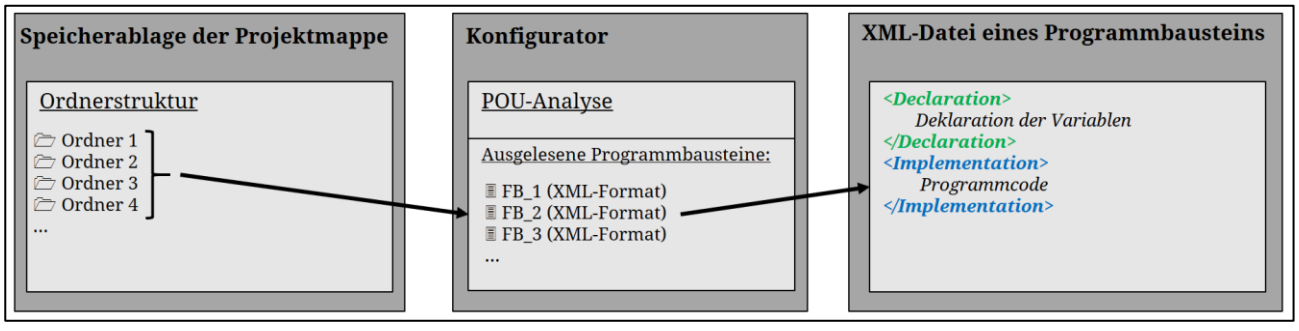

Bild 2: Aufbau der Ordnerstruktur eines Steuerungsprogramms

Der Aufbau der Ordnerstruktur eines Steuerungsprogramms ist in Bild 2 dargestellt. Anders als im PLCopenXML-Format werden hier nicht alle Projektdaten in einem XML-Dokument zusammengefasst, sondern auf mehrere Unterverzeichnisse verteilt. Deshalb werden die einzelnen Ordner der Projektmappe vom Konfigurator automatisiert durchsucht, um die im Steuerungsprogramm verwendeten Programmorganisationseinheiten (POUs) bzw. Programmbausteine zu identifizieren. Viele Steue-

rungshersteller, insbesondere jene mit hohem Marktanteil, greifen zum Abspeichern der Programmbausteine auf ein XML-Format zurück. Die Analyse dieser XML-Dokumente zeigt, dass der Aufbau des XML-Dokuments unabhängig von dem darin implementierten Programmcode identisch bleibt. Dies erleichtert die Navigation eines Textparsers durch das XML-Dokument. In Bild 2 sind die XML-Knoten grün und blau hinterlegt, welche zur Konfiguration einer Test-Applikation von Bedeutung sind. Im Knoten <*Declaration*> werden die Variablendeklarationen hinterlegt, welche direkt im Programmbaustein deklariert werden. Im Knoten <*Implementation*> wird der gesamte Programmcode abgespeichert, welcher letztendlich die programmspezifischen Funktionalitäten des Steuerungsprogramms definiert. Durch das gezielte Zugreifen auf den Inhalt der beschriebenen XML-Knoten können somit Variablendeklarationen sowie Programmfunktionalitäten des Programmbausteins automatisiert identifiziert werden. Aus diesen Informationen kann anschließend die Test-Applikation zur virtuellen Inbetriebnahme konfiguriert werden. Somit bietet die Analyse der Unterverzeichnisse eines Projektes eine bessere Möglichkeit die Informationen zur virtuellen Inbetriebnahme zu erfassen, als die Analyse eines PLCopenXML-Dokuments. Die Analysen zur Konfiguration einer Test-Applikation werden im Folgenden genauer beschrieben.

3.2.1 Analyse der Variablendeklaration

Zur Analyse der im Steuerungsprogramm verwendeten Variablen ist der Deklarationsbereich der Programmbausteine zu betrachten. Eine Variablendeklaration definiert die Eigenschaften einer Variable innerhalb des Steuerungsprogramms. Hierbei wird u. a. die Variablenbezeichnung, physikalische Adresse, Schnittstelleneigenschaften, der Datentyp und Kommentare zur Beschreibung der Variable festgelegt. Durch den Konfigurator müssen diese Variableneigenschaften identifiziert und in die Test-Applikation übertragen werden, um die Grundlage einer Schnittstelle zum Steuerungsprogramm zu schaffen. Ein beispielhafter Analyseprozess sowie die Bearbeitung der Variablen zur Implementierung in die Test-Applikation ist in Bild 3 dargestellt.

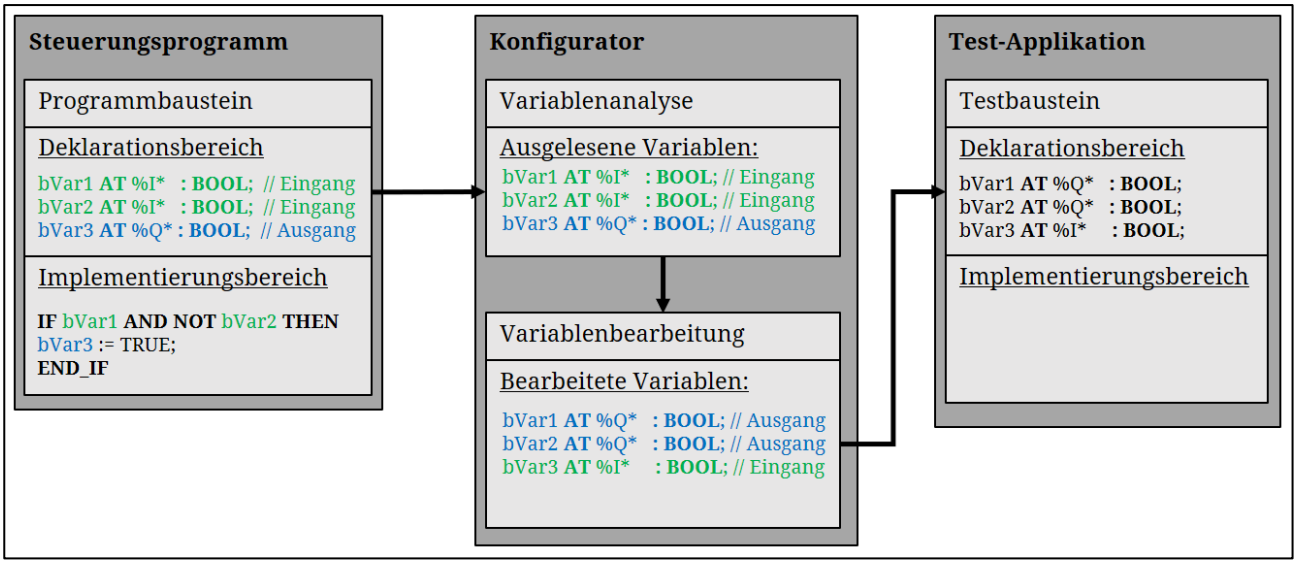

Bild 3: Analyse und Bearbeitung der Variablendeklarationen

Prinzipiell müssen Ausgangssignale einer Steuerung von einer Testanwendung eingelesen und ausgewertet werden. Dazu gehören Diagnosemeldungen der Anlage, Prozessinformationen, Ansteuerungen von Feldgeräten, etc. Eingangssignale der Steuerung hingegen müssen von der Test-Applikation erzeugt und von dieser ausgegeben werden. Dazu gehören Rückmeldungen von Feldgeräten, Störmel-

dungen, Sensordaten, etc. Der Konfigurator liest daher zuerst die Variablendeklarationen der Programmbausteine ein. Anschließend werden, wie in Bild 3 dargestellt, alle Eingangsvariablen des Steuerungsprogramms in Ausgangsvariablen umgewandelt und alle Ausgangsvariablen in Eingangsvariablen, um ein Spiegelbild des Prozessabbilds des Steuerungsprogramms zu erhalten. Abschließend werden die Variablendeklarationen in den Programmbaustein einer Test-Applikation geschrieben, welcher letztendlich die Testszenarien abarbeitet.

3.2.2 Analyse zur Generierung der Testszenarien

Die Variablendeklaration beinhaltet die Schnittstelleninformation und der Implementierungsbereich die Information über die Programmfunktionalität. Die Herausforderung besteht nun darin, aus weitestgehend freiformulierten Programmen die Funktionalität einer Test-Applikation abzuleiten, da für eine Programmfunktion mehrere Varianten der Implementierung bestehen. Beispielsweise kann die IF-Bedingung im Implementierungsbereich des Steuerungsprogramms in Bild 4 ebenfalls durch eine SWITCH-CASE-Struktur programmiert werden. Eine Möglichkeit zur Ableitung von Testszenarien bietet die Identifikation von grundlegenden Ausführungsstrukturen wie WENN-DANN oder SWITCH-CASE-Strukturen, logischer Operatoren oder Schleifen. Diese haben immer den gleichen Aufbau. Hierbei müssen Bedingungen erfüllt werden, welche in einer Ausführungsanweisung münden. Daraus lassen sich bereits Testfunktionalitäten ableiten. Des Weiteren können Testszenarien unter Verwendung von Kommentaren bei der Programmentwicklung definiert werden. Optional kann eine Kombination beider Ansätze umgesetzt werden. Aufgrund des umfangreichen Programmieraufwands der Analyse aller Ausführungsstrukturen im Programmcode, wird im Folgenden der Ansatz der Kommentarzeilen während der Programmentwicklung verfolgt. Der Vorteil hierbei ist, dass die Testbedingungen durch definierte Schlüsselwörter in den Kommentarzeilen repräsentiert und durch den Konfigurator identifiziert werden können. Allerdings wird hier ein zusätzlicher Programmieraufwand bei der Erstellung des Steuerungsprogramms ersichtlich. Die Kommentare müssen die wesentlichen Informationen der Testszenarien beinhalten. Grundlegend sind Testszenarien durch eine Testbedingung, wie z. B. logische Verknüpfungen von Variablen oder Eingangssignalen und einem Testergebnis, also den Zustand von Ausgangsvariablen des Steuerungsprogramms, definiert [9]. Ein Beispiel solcher Kommentarzeilen ist im Implementierungsbereich des Steuerungsprogramms in Bild 4 abgebildet.

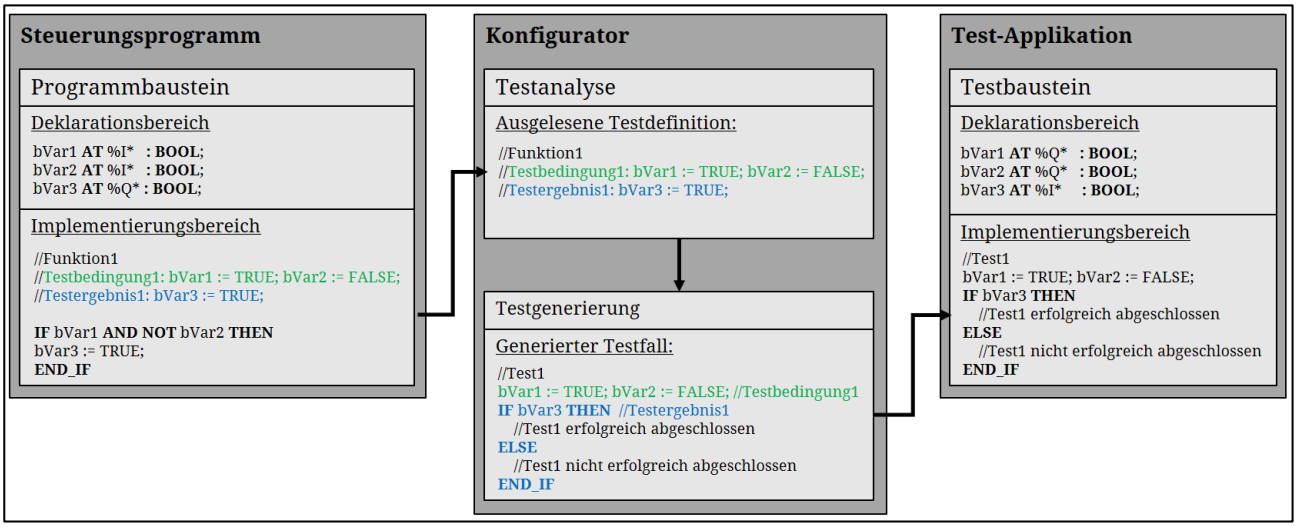

Bild 4: Analyse des Implementierungsbereichs und Generierung der Testfälle

Durch den Kommentar *//Testbedingung1* wird dem Konfigurator signalisiert, dass in dieser Codezeile die Testbedingung des ersten Funktionstests definiert ist. Der Kommentar *// Testergebnis1* signalisiert

das erwartete Testergebnis des ersten Tests. Durch das Auslesen dieser Codezeilen durch den Konfigurator kann der Testfall abgeleitet werden. Zu Beginn des Tests wird die Testbedingung implementiert, also die Eingangsvariablen des Steuerungsprogramms beschrieben. Anschließend wird abgefragt, ob das erwartete Testergebnis erfüllt ist. Dies wird in Form einer IF-Abfrage durchgeführt. Der durch den Konfigurator definierte Testcode wird anschließend in den Implementierungsbereich der Test-Applikation überführt.

3.3 Abarbeitung der Testszenarien

Nachdem die Testszenarien vom Konfigurator erzeugt wurden, müssen diese anschließend automatisiert abgearbeitet werden. Hierfür eignet sich u. a. eine Schrittkette im Testbaustein der Testapplikation. Demzufolge fügt der Konfigurator den generierten Testcode in den entsprechenden Testschritt der Schrittkette ein. Die Reihenfolge der Testszenarien kann entweder durch die Reihenfolge der Funktionalitäten im Steuerungsprogramm, durch Priorisierung gewisser Operatoren oder durch das Einfügen der Testnummern innerhalb von Kommentarzeilen festgelegt werden. Neben der Abarbeitung der Testschritte sind weitere Dokumentationsaufgaben während der Testdurchführung zu berücksichtigen. Hierfür wird ein weiterer Programmbaustein in der Test-Applikation entwickelt. Dieser trägt alle Testergebnisse, wie z. B. Erfolg der Testdurchführung, Reaktionszeit oder allgemeine Informationen zur Testdurchführung, wie Datum, Uhrzeit, etc., zusammen und schreibt diese in ein Prüfprotokoll. Dieses kann beispielsweise in Form eines CSV-Dokuments exportiert werden. Ein Beispiel einer Schrittkette ist im Implementierungsbereich des Testbausteins in Bild 5 dargestellt. Der Programmbaustein mit der Bezeichnung *fb_Doku* wird immer nach der Testüberprüfung aufgerufen. Mit einer Eingangsvariable *bTestKorr* wird dem Baustein mit einem *TRUE*-Wert ein erfolgreicher Test signalisiert, während ein *FALSE*-Wert ein fehlerhaftes Testergebnis bedeutet. Somit erstellt der Baustein ein CSV-Dokument, welches die Testergebnisse beinhaltet. Nach Abschluss des Tests und der Testdokumentation wird in den nächsten Testschritt gewechselt.

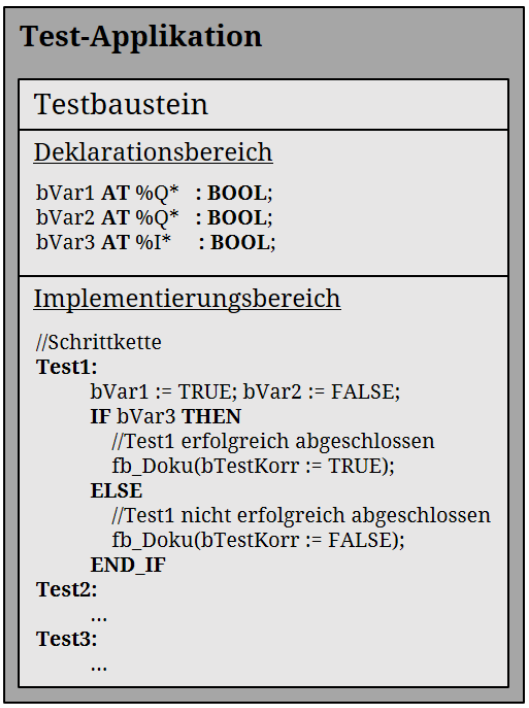

Bild 5: Abarbeitung der Testszenarien in der Test-Applikation

3.4 Simulation von Instanzen des Steuerungsprogramms

Nach der Definition der Abarbeitung der Testszenarien kann die Test-Applikation bereits an einer realen Anlage angewendet werden. Allerdings ist bei einer virtuellen Inbetriebnahme noch keine Hardware vorhanden, welche Signale an das Steuerungsprogramm liefert. Deshalb müssen diese Signale von Simulationsbausteinen in der Test-Applikation generiert werden. In einem Simulationsbaustein muss die Verhaltensweise eines Feldgeräts und die Reaktionen auf bestimmte Eingangssignale softwaretechnisch umgesetzt werden. Bislang finden sich keine Konzepte und Untersuchungen, aus Steuerungsprogrammen Simulationsbausteine autonom abzuleiten. Dies wird daher als Ausblick erachtet. Jedoch lassen sich wiederkehrende Steuerungskomponenten und deren Funktionalität in Bibliotheken anlegen, die nach erstmaliger Generierung zukünftigen Projekten zur Verfügung gestellt werden. Diese können wiederum autonom vom Konfigurator genutzt werden. Zur Analyse, welche Simulationsbausteine zum Testen eines Steuerungsprogramms benötigt werden, werden vergleichbar mit der Analyse zur Generierung der Testszenarien in Abschnitt 3.2.2 zusätzliche Kommentare im Programmcode eingefügt. Diese werden im Deklarationsbereich vor der Instanziierung des Bausteins eingefügt.

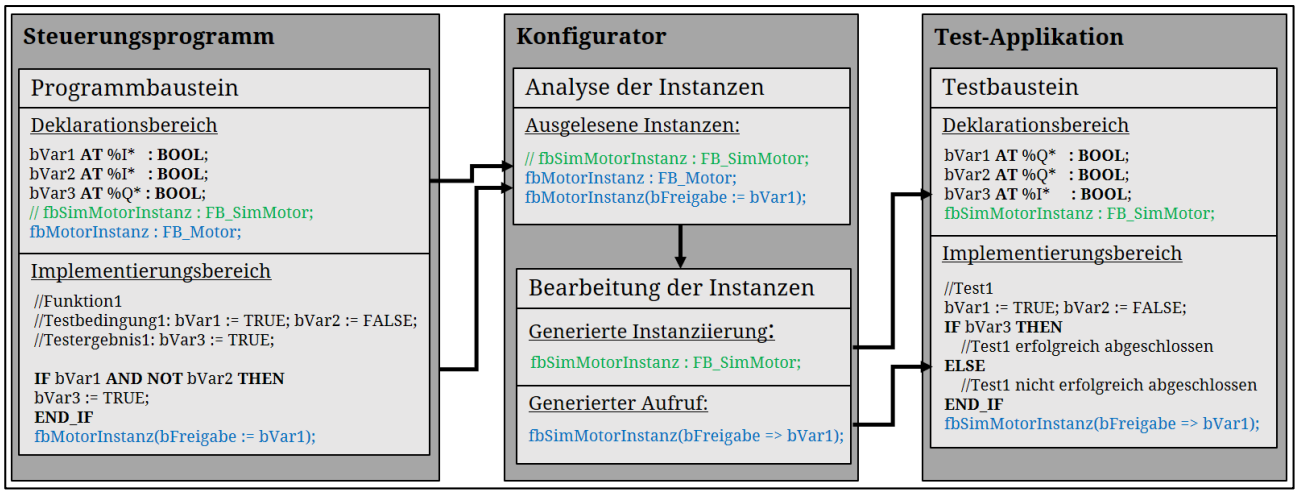

Bild 6: Einbindung eines Simulationsbausteins in den Testprozess

Ein Beispiel zur Einbindung von Simulationsbausteinen in den Testprozess ist in Bild 6 dargestellt. Im Steuerungsprogramm wird im Deklarationsbereich eine Instanz *fbMotorInstanz* vom Typ *FB_Motor* angelegt. Der Kommentar in der vorangehenden Zeile definiert den Simulationsbaustein, welcher in der Test-Applikation verwendet werden soll. Im Implementierungsbereich des Steuerungsprogramms wird der Baustein *fbMotorInstanz* aufgerufen und seiner Eingangsvariable *bFreigabe* der Wert der Eingangsvariable bVar1 zugewiesen. Nachdem der Konfigurator die Test-Applikation wie oben beschrieben konfiguriert hat, ist im Deklarationsbereich des Testbausteins die Instanziierung des Simulationsbausteins *fbSimMotorInstanz* vom Typ *FB_SimMotor* zu erkennen. Im Implementierungsbereich des Testbausteins wird der Simulationsbaustein aufgerufen. Die Eingangsvariable der Instanz im Steuerungsprogramm wurde im Simulationsbaustein zu einem Ausgang konvertiert und die Variable bVar1 wird nun vom Simulationsbaustein beschrieben.

3.5 Schnittstellen zum Steuerungsprogramm

Zur Kommunikation zwischen Test-Applikation und Steuerungsprogramm wird eine geeignete Schnittstelle benötigt. Im Rahmen dieser Arbeit wurde hierzu das Kommunikationsprotokoll OPC UA ausgewählt. Dieses hat den Vorteil, dass ein virtueller OPC UA Server auf dem Entwicklungsrechner initialisiert werden kann. Der Weiteren ist das Kommunikationsprotokoll OPC UA standardisiert und somit unabhängig von der Entwicklungsumgebung einsetzbar [10]. Zur virtuellen Inbetriebnahme

muss somit das Steuerungsprogramm mit einem OPC UA Sever verbunden werden, während die Test-Applikation mit einem virtuellen OPC UA Client ausgestattet wird. Somit kann von der Test-Applikation auf das Prozessabbild des Steuerungsprogramms zugegriffen werden. Allerdings sind hier ebenso andere Kommunikationsformen, wie z. B. TCP/IP, MQTT oder ADS realisierbar [11], [12], [13]. Im weiteren Verlauf der Entwicklung eines Konfigurators zur virtuellen Inbetriebnahme müssen deshalb unterschiedliche Kommunikationsformen untersucht und verglichen werden. Bild 7 verdeutlicht die Kommunikation zwischen Test-Applikation und Steuerungsprogramm während der virtuellen Inbetriebnahme.

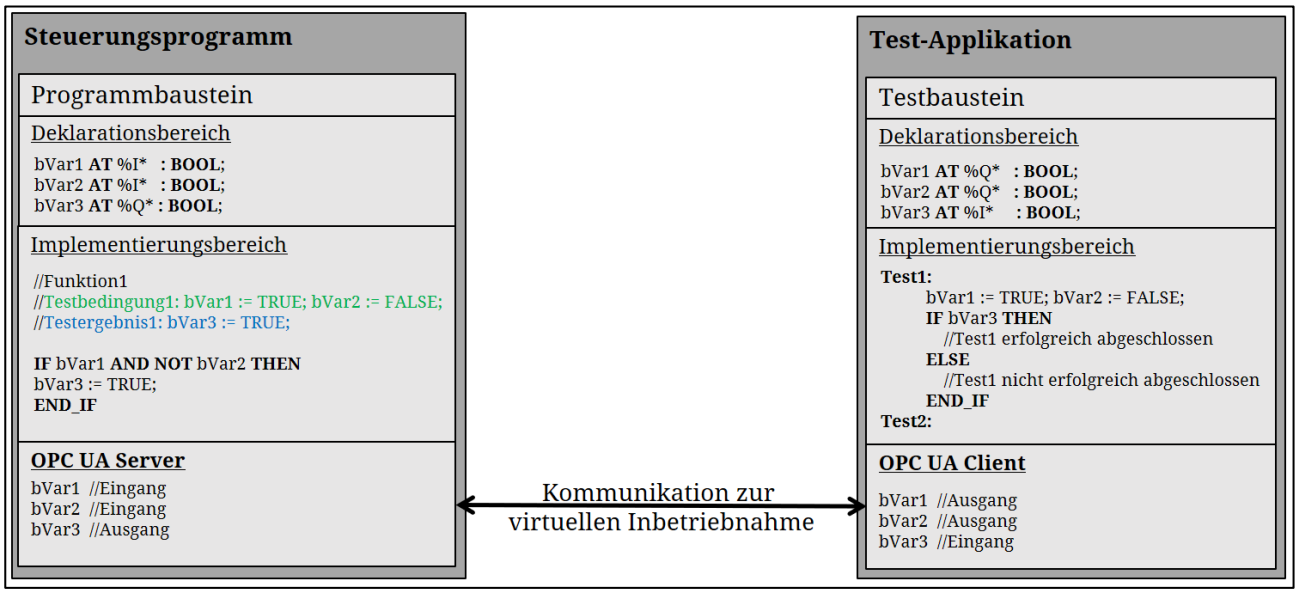

Bild 7: Beispiel einer Schnittstelle zur virtuellen Inbetriebnahme

4 Ablauf der automatisierten virtuellen Inbetriebnahme

Nach der Entwicklung eines Konzeptes zur automatisierten Konfiguration von Testszenarien und deren Abarbeitung durch eine Test-Applikation besteht nun die Herausforderung darin, ein Konzept zum Ablauf der automatisierten virtuellen Inbetriebnahme aufzustellen. Ein möglicher Ablauf ist in Bild 8 dargestellt. Hierbei ist gleichzeitig der Aufwand zur Bedienung des Konfigurators und der Test-Applikation während der virtuellen Inbetriebnahme zu erkennen.

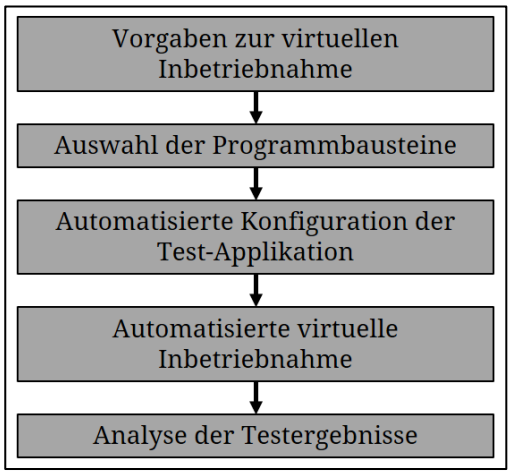

Bild 8: Möglicher Ablauf der automatisierten virtuellen Inbetriebnahme

Zur Vorbereitung einer virtuellen Inbetriebnahme wird dem Konfigurator das Steuerungsprogramm, welches im Folgenden getestet werden soll, vorgegeben. Diese Vorgabe kann durch den Speicherpfad erfolgen, der den Speicherort der Projektmappe definiert. Alternativ kann dem Konfigurator der Dateiname des Steuerungsprogramms übergeben werden, nach welchem der Speicher durchsucht werden kann. Dadurch wird dem Konfigurator die Identifizierung der POUs im Steuerungsprogramm ermöglicht. Nach dieser automatisierten Identifizierung werden die Bezeichnungen der POUs ausgegeben, um eine Auswahl durch den Bediener zu ermöglichen, welcher Baustein im Test einbezogen werden soll. Daher können gezielt Bausteine vom Test ausgeschlossen werden. Dieser Vorteil ist besonders in der Entwicklungsphase eines Steuerungsprogramms zu erkennen, da die Funktionalitäten einzelner Programmbausteine bereits in der Entwicklungsphase getestet werden können, auch wenn das Steuerungsprogramm nicht vollständig fertiggestellt ist. Anschließend wird die automatisierte Konfiguration der Test-Applikation aktiviert. Nach Abschluss dieses Prozesses beinhaltet die Test-Applikation die vollständigen Testinformationen. Nun kann die virtuelle Inbetriebnahme durchgeführt werden. Nach Abschluss der virtuellen Inbetriebnahme wird ein Testprotokoll ausgegeben. Wurden fehlerhafte Testergebnisse dokumentiert, muss eine Fehlerkorrektur dieser Fehlfunktionen im Steuerungsprogramm durch den Programmentwickler durchgeführt werden. Anschließend kann die virtuelle Inbetriebnahme erneut durchgeführt werden. Sind keine fehlerhaften Testergebnisse aufgetreten, wird die virtuelle Inbetriebnahme abgeschlossen.

5 Ausblick

Die ersten Tests haben gezeigt, dass es mit Hilfe des Konfigurators unter Verwendung des beschriebenen Konzepts möglich ist, die Informationen zur virtuellen Inbetriebnahme aus einem Projekt automatisiert auszulesen, zu bearbeiten und anschließend in eine Test-Applikation zu überführen. Im weiteren Entwicklungsprozess des Konfigurators wird geprüft, wie weit das Konzept zur Testanalyse auf unterschiedliche Steuerungsprogramme, vor allem auf umfangreichere und komplexere Programme übertragbar ist. Des Weiteren werden unterschiedliche Schnittstellen zwischen dem Steuerungsprogramm und der Testapplikation untersucht, um einen idealerweise echtzeitfähigen Signalfluss während der virtuellen Inbetriebnahme sicherstellen zu können. Weiterhin sollen verschiedene Projektdaten, wie z. B. CAD-Pläne oder Funktionsbeschreibungen, als zusätzliche Testbasen zur Analyse durch den Konfigurator herangezogen werden und in den Konfigurationsprozess der Test-Applikation eingebunden werden. In diesem Zusammenhang kann die EG-Richtlinie ebenfalls Berücksichtigung finden. Beispiele hierfür sind in Bild 9 dargestellt.

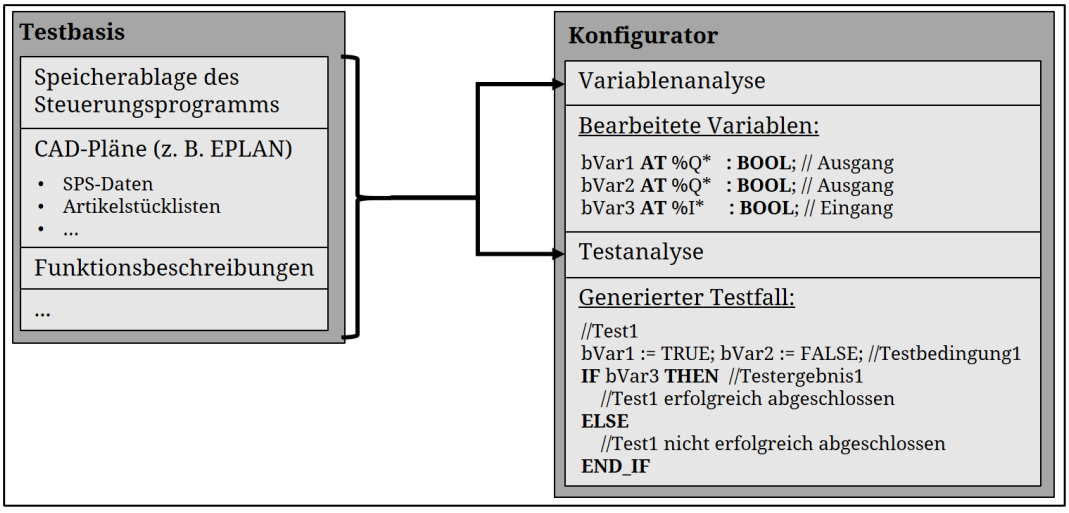

Bild 9: Einbeziehung mehrerer Testbasen zur Konfiguration einer virtuellen Inbetriebnahme

Abschließend werden Praxistests unter Einbeziehung von Probanden durchgeführt, um den Zeitaufwand und die Qualität der virtuellen Inbetriebnahme unter Verwendung des Konfigurators zu bewerten. Durch diese Praxistests kann eine Aussage über die Bedienbarkeit des Konfigurators getroffen und die Wirtschaftlichkeit des Konfigurators vergleichend untersucht werden.

6 Literatur

[1] Rauen, H.;Mosch, C.; Axmann, E.: Leitfaden virtuelle Inbetriebnahme, VDMA Verlag GmbH, Frankfurt am Main, 2020.
[2] ISG-Stuttgart: ISG virtuos – Virtuelle Inbetriebnahme: https://www.isg-stuttgart.de/de/isg-virtuos/virtuelle-inbetriebnahme.html (Zugriff: 13.08.2021)
[3] Steirer, T.: Basiswissen Testautomatisierung, 2. Aufl., dpunkt.verlag, Heidelberg, 2015.
[4] codekickers.news, IEC 61131-3: Untit-Tests, http://codekicker-.de/news/IEC-61131-3-Unit-Tests (Zugriff: 29.03.2021).
[5] Siemens: Dokumentation SIMATIC S7-PLCSIM Advanced/STEP 7, 03/2018.
[6] Spitzweg, M.: Dissertation: Methode und Konzept für den Einsatz eines physikalischen Modells in der Entwicklung von Produktionsanlagen, Herbert Utz Verlag, München, 2009.
[7] Steirer, T.: Basiswissen Testautomatisierung, 2. Aufl., dpunkt.verlag, Heidelberg, 2015.
[8] PLCopen: XML-Exchange: https://plcopen.org/technical-activities/xml-exchange (Zugriff 11.05.2021).
[9] Spillner, A.; Linz, T.: Basiswissen Softwaretest, 6. Aufl., dpunkt.verlag GmbH, Heidelberg, 2019.
[10] Siemens: OPC UA – Open Platform Communiations Unified Architecture: https://new.siemens.com/de/de/produkte/automatisierung/industrielle-kommunikation/opc-ua.html?gclid=Cj0KCQiA7oyNBhDiARIsADtGRZZJ3Tw_r6rnMYg3jdHr-veTDZa_qKCMCxh2i7XZi-FZaary1GGUSNkaAlO1EALw_wcB (Zugriff: 29.11.2021).
[11] Spitzweg, M.: Dissertation: Methode und Konzept für den Einsatz eines physikalischen Modells in der Entwicklung von Produktionsanlagen, Herbert Utz Verlag, München, 2009.
[12] MQTT: Why MQTT?: https://mqtt.org/ (Zugriff: 26.11.2021).
[13] Beckhoff: Einführung ADS: https://infosys.beckhoff.com/index.php?content=../content/1031/tcadscommon/html/tcadscommon_introads.htm&id= (Zugriff: 26.11.2021).