Vollständiger Test des Automatisierungssystems

Domänenübergreifende Integration heterogener Hardware-Emulatoren in einem virtuellen Framework

Dipl.-Ing. (FH) Mario Hoernicke, Dr.-Ing. Jürgen Greifeneder,
Dr.-Ing. Mike Barth; ABB Forschungszentrum, Ladenburg

Kurzfassung

Neben dem klassischen „Signal-Forcing“ hat sich Simulation als Mittel zum Testen der Leitsystem-Konfiguration etabliert. Aufgrund des hohen, für die Erstellung der Simulationen notwendigen, manuellen Aufwandes fokussieren simulationsbasierte Testmethoden meist auf den funktionalen Bereich, wobei ausschließlich die Korrektheit der Steuerungslogik, nicht aber die Konfiguration der Subsysteme geprüft wird. Im vorliegenden Beitrag wird eine Methode vorgestellt, mittels derer die Vorteile simulationsbasierter Tests um die Möglichkeit moderner Emulatoren für Steuerungen und deren Peripherie erweitert werden. Hierzu wird ein Framework vorgestellt, welches die für einen vollständigen Test notwendigen Emulatoren der Automatisierungs- und zugehöriger Subsysteme integriert und automatisch konfiguriert.

1. Motivation

Im Engineering von Automatisierungssystemen für Anlagen der Prozessindustrie gewinnt die Phase des Testens an immer größerer Bedeutung. Je komplexer die Anlagen werden, desto umfangreichere Tests werden benötigt, um die Korrektheit des Automatisierungssystems im Sinne der Kundenspezifikation zu überprüfen. Angelehnt an die Testprozeduren von Software-Applikationen beinhaltet das Engineering eines Prozessleitsystems mehrere Prüfschritte. Je nach Fortschritt bei der Implementierung der Leitsystemfunktionen werden Grenzwerte für Alarme und Meldungen, Verriegelungen, Schrittketten sowie Wirkrichtungen von Reglern darauf hin überprüft, ob sie den Spezifikationen entsprechen. Die Ingenieure stehen hierbei vor der Herausforderung, dass die für den FAT (engl. Factory Acceptance Test) zur Verfügung stehende Zeit immer kürzer wird (Reduzierung der Gesamtprojektdauer seit 1970: 25 % [1]), wogegen die Anzahl und Komplexität der durchzuführenden Tests aufgrund des zunehmenden Funktionsumfangs der Automatisierung stetig zunimmt. Der FAT wird jedoch durch das Fehlen der automatisierungstechnischen Komponenten (wie beispiels-

weise reale Steuerungs-Hardware oder reale Feldbuskomponenten) im Prüffeld erschwert. Die Peripherie-Komponenten werden aus logistischen Gründen oft direkt zur Anlage versandt. Das Prüffeld besteht deshalb oft aus Beispiel-Hardware-Komponenten, welche ausschließlich einen Teiltest des Automatisierungssystems erlauben [2]. Die Kommunikation zwischen Peripherie und Leitsystem kann daher nur unvollständig getestet werden. Zusätzlich zu der steigenden Komplexität, bewirken weitere Aspekte wie der steigende Kosten- und Zeitdruck sowie ein geographisch verteilter Engineeringprozess, dass der FAT zu einer signifikant wichtigen Engineering Phase für die Qualität des AT-Systems wird, welche trotz allem keinen vollständigen Systemtest erlaubt.

Für einen effizienteren FAT werden zunehmend etablierte Methoden der virtuellen Inbetriebnahme [3] eingesetzt. Hierbei wird der zu automatisierende Prozess mittels geeigneter Software simuliert, wobei die Steuerung des Automatisierungssystems entweder real vorhanden ist oder emuliert wird. Mit letzterer Vorgehensweise kann die Steuerungslogik auch ohne das Vorhandensein realer Hardware getestet werden.

Um bei simulationsbasierten Tests nicht ausschließlich auf den funktionalen Bereich zu fokussieren, werden unlängst Emulatoren für Subsysteme und die Peripherie der Steuerungen eingesetzt. Oft ist es die Peripherie des Automatisierungssystems welche den größten Aufwand im Rahmen der Testphasen – insbesondere während des FAT – erzeugt. Wie in Bild 1 dargestellt, existieren Emulatoren für einfachere E/A-Applikationen (E/A-Emulator) bis hin zu komplexen Emulatoren für Feldbussysteme. Beispiele sind Emulatoren für elektrische Verteilsysteme [4] auf Basis des IEC61850 Standards [5], bzw. Emulatoren für eine auf Foundation Fieldbus (FF) basierende dezentrale Steuerungslogik [6].

Da die Steuerungslogik zunehmend auf intelligente Feldgeräte verteilt wird (vgl. [7]), bedeutet der Einsatz von HW-Emulatoren für Tests des Automatisierungssystems einen wichtigen Schritt in Richtung einer höheren Testabdeckung. Der Fokus dieses Beitrags wird daher auf die korrekte Emulation des Automatisierungssystems gesetzt. Die darüber hinausgehende Notwendigkeit der Prozesssimulation wird in [8] und [9] beschrieben.

Trotz der erhöhten Testabdeckung werden Emulatoren bislang nur selten eingesetzt. Der Grund hierfür liegt im hohen manuellen Aufwand für die Parametrierung und Konfiguration der Emulations-Software. Des Weiteren wird ein beträchtlicher Aufwand für Bereitstellung, Administration und Konfiguration der benötigten IT-Infrastruktur notwendig.

Um eine einheitliche und automatisierte Konfiguration - bestehend aus Emulatoren und umgebender IT-Infrastruktur - zu ermöglichen, wird in diesem Beitrag das Konzept eines

„Virtuellen Emulatoren Frameworks“ (im Folgenden als „Framework“ bezeichnet) vorgestellt. Mithilfe dieses Konzepts wird eine signifikante Effizienzsteigerung in der Vorbereitung der Emulation für den ganzheitlichen Test von Automatisierungssystemen erreicht.

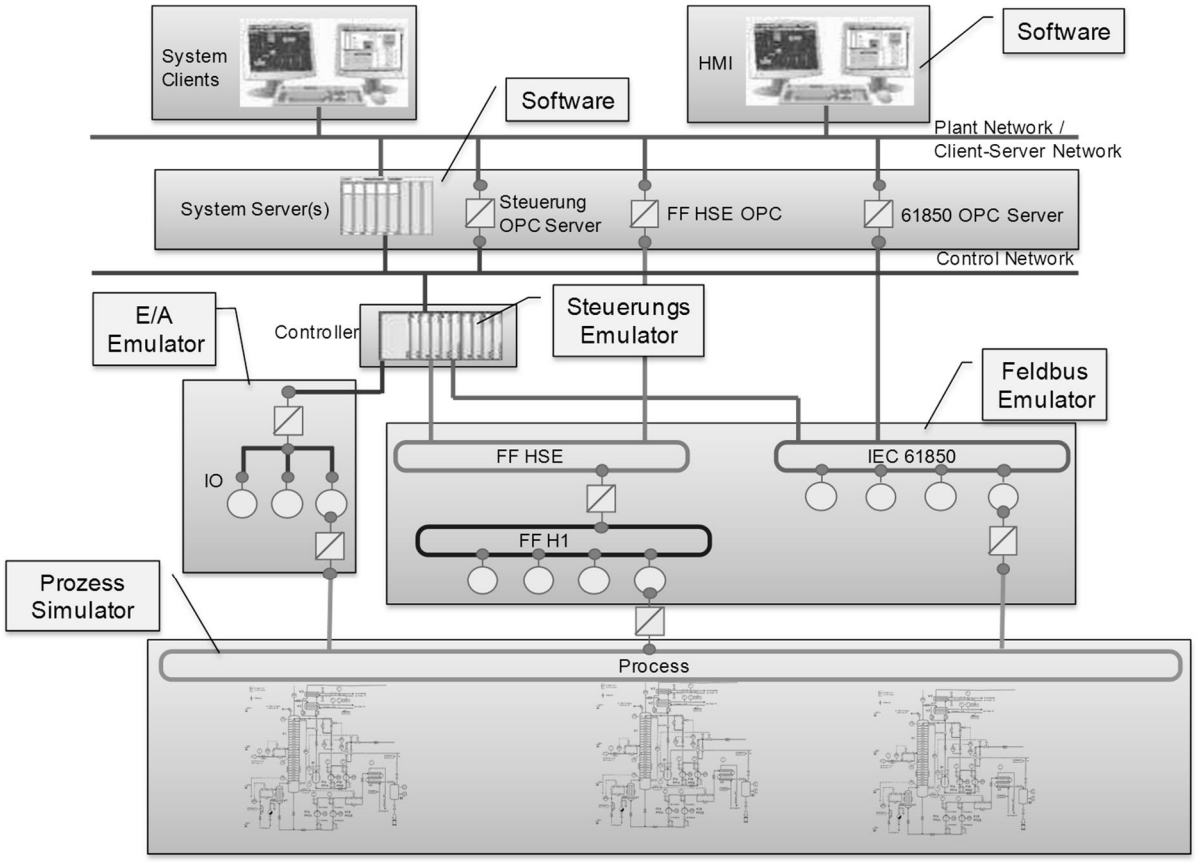

Bild 1: Virtuelle „Hardware-in-the-Loop“ Testumgebung

2. Methodische Grundlagen zu Emulatoren für Automatisierungssysteme

Der grundlegende Ansatz der Emulationstechnologie ist nach [10] das Verhalten eines Geräts mittels eines anderen Geräts oder einer Software nachzubilden. Dieser Ansatz wird in unterschiedlichen Branchen verwendet um Systemtests (z.B. Funktions- und Integrationstests) durchführen zu können, bevor die benötigten Hardwarekomponenten zur Verfügung stehen. So werden Emulatoren beispielsweise bei der Entwicklung eingebetteter Systeme verwendet um das Verhalten von Mikrocontrollen nachzubilden. Hierbei werden die Firmware sowie die Anwendungssoftware getestet, bevor die reale Hardware in Produktion geht.

Auch in der Automatisierungstechnik werden Emulatoren zunehmend eingesetzt und zur virtuellen Inbetriebnahme verwendet. Um die Steuerungslogik einer SPS (Speicherprogrammierbare Steuerung) zu testen, werden u.a. SPS-Emulatoren (SoftSPS) eingesetzt, auf wel-

che in IEC 61131-3 [11] implementierte Applikationen ausgeführt werden können. Des Weiteren werden Emulatoren für Feldbus-Systeme eingesetzt, um die Logik [6] sowie das zeitliche Verhalten des Busses [12] realitätsnah zu testen. Hinzu kommen Emulatoren zur Nachbildung eines einfacheren E/A-Verhaltens von Feldkomponenten sowie Emulatoren für Feldbus-Koppler.

Die wachsende Vielfalt von Emulatoren in der Automatisierungstechnik erschwert jedoch die Vorbereitung der Tests, da jeder Emulator speziell konfiguriert und parametriert werden muss. Bei großen Systemen, mit verschiedenen Subsystemen und komplexer Peripherie, müssen die Emulationslösungen exakt aufeinander abgestimmt sein. Oftmals entsteht durch die Konfiguration und Verknüpfung der Emulatoren ein Netzwerk, welches dem Komplexitätsniveau des realen AT-Systems nahe kommt. Dies resultiert in einem, im Vergleich zum Teilsystemtest, deutlichen Mehraufwand für den FAT [13].

Dabei ist der für die Ausführung der Emulation notwendige Konfigurationsaufwand der IT-Infrastruktur noch nicht berücksichtigt: Die meisten Emulatoren benötigen eine separate Ethernet-Schnittstelle, mit einer auf den Emulator abgestimmten IP-Adresse. Einige Emulatoren sind zudem nicht multi-Instanz-fähig, weshalb mehrere Instanzen des gleichen Typs auf verschiedenen Rechnern verteilt emuliert werden müssen. Der manuelle Aufwand hierfür erlaubt kein positives Kosten-Nutzen-Verhältnis. Zudem nimmt der Aufwand mit der Anzahl der verwendeten Subsysteme, und gleichbedeutend mit der steigendenden Komplexität des Automatisierungssystems, zu.

2.1 Anforderungen an ein Framework

Das „Virtuelle Emulator Framework“ soll die Administration der Rechnerinfrastruktur sowie das Entwerfen der Emulation vereinfachen und automatisieren. Um dies zu erreichen ist es wichtig, die grundlegenden Anforderungen an ein solches Framework bzgl. dessen Nutzung und Administration zu kennen. Basierend auf dem zu erzielenden Nutzen lassen sich vier grundlegende Anforderungen an das Emulator Framework formulieren:

1. Automatische Konfiguration der Emulatoren: Jeder Emulator benötigt die Konfiguration der zu emulierenden Komponente (z.B. eines FF-Subnetzes). Das Framework muss in der Lage sein, diese Konfiguration - mit Hilfe des integrierten Emulators - automatisch zu erzeugen und dem Emulator bekannt zu geben.
2. Automatische Konfiguration der IT-Infrastruktur: Die IT-Infrastruktur muss auf die darauf auszuführenden Emulatoren abgestimmt sein. Hierbei gilt es, die benötigten Ressourcen automatisch bereitzustellen und die notwendige Konfiguration zu automatisieren. Folglich

wird eine skalierbare Infrastruktur benötigt, die, je nach Bedarf, Ressourcen zur Verfügung stellen kann.

3. Einheitliches Bedienkonzept: Das Framework soll ein einheitliches Bedienkonzept besitzen. Die Bedienoberfläche - welche zur Administration der IT-Infrastruktur, sowie der Auswahl der zu emulierenden Komponenten dient - soll unabhängig von den integrierten Emulatoren sein.
4. Keine Änderungen der Onlinekommunikation der Emulatoren: Die Kommunikationsprotokolle der Emulatoren dürfen nicht beeinflusst oder verändert werden, da diese spezifisch auf die verwendeten Engineering-Werkzeuge und das zu emulierende System abgestimmt sind. Eine Vereinheitlichung der Kommunikation während der Laufzeit ist unzulässig und würde zusätzlichen Aufwand bedeuten.

Die formulierten Anforderungen stehen im engen Zusammenhang mit den Emulatoren sowie deren Konfiguration und Funktionsweise. Aus diesem Grund wird im folgenden Abschnitt die Funktionsweise in Bezug auf die Konfiguration von Emulatoren der Automatisierungstechnik vorgestellt. Stellvertretend wird hierfür der Foundation Fieldbus Emulator nach [6] verwendet und das Vorgehen zu dessen Konfiguration erläutert.

2.2 Emulatoren in der Automatisierungstechnik

Die Funktionsweise von Emulatoren für automatisierungstechnische Komponenten basiert auf immer gleichen Prinzipien. Zunächst muss der Emulator die Konfiguration der zu emulierenden realen Komponente kennen. Hierfür wird eine Konfiguration benötigt, welche auf eine der beiden nachfolgend beschriebenen Möglichkeiten aufgesetzt werden kann:

1. Die Konfiguration eines einzelnen Geräts, z.B. einer SPS, kann mittels eines Downloads der IEC61131-3 [10] Applikation auf die SoftSPS durchgeführt werden. Diese Variante wird üblicherweise eingesetzt, wenn ein separates Gerät durch eine Instanz des Emulators emuliert wird. Dies geschieht über die gleichen Kommunikationsprotokolle wie beim realen System.
2. Eine Konfigurationsdatei wird aus Engineering-Daten erzeugt und zur Konfiguration des Emulators verwendet. Diese Möglichkeit wird üblicherweise zur Konfiguration einer Emulation für Subsysteme verwendet, bei denen die Topologie des Subsystems dem Emulator vor der Konfiguration nicht bekannt ist.

2.2.1 Konfiguration einer Foundation Fieldbus Emulation

Ein Beispiel für einen Emulator mit Datei-basierter Konfiguration ist der Foundation Fieldbus Emulator nach [6]. Um eine FF-Emulation vorzubereiten sind die folgenden Schritte nötig:

Schritt 1: Auswahl des zu emulierenden Subnetzes: Eine FF Lösung besteht üblicherweise aus mehreren voneinander getrennten Subnetzen. Der referenzierte Emulator ist in der Lage, in einer Instanz ein Subnetz zu emulieren. Dies bedeutet, dass der Nutzer zunächst die zu emulierenden Subnetze auswählen muss.

Schritt 2: Erzeugen der Konfigurationsdatei: Als nächstes wird eine Konfiguration des ausgewählten Subnetzes benötigt. Hierfür wird ein Exporter verwendet, welcher auf die Daten im Engineering-System zugreift und diese in ein Datei-basiertes Zwischenformat exportiert, da die Emulation üblicherweise nicht auf dem Engineering-PC ausgeführt wird.

Schritt 3: Bereitstellen einer Ethernet Schnittstelle auf dem Emulationsrechner: FF ist ein Feldbus, welcher ein Ethernet-basiertes Protokoll (High Speed Ethernet, kurz HSE) als Verbindung zum Leitsystem sowie zu den verteilten Steuerungen verwendet. Für jede Instanz des Emulators, d.h. für jedes zu emulierende Subnetz, muss dementsprechend eine Ethernet-Schnittstelle verfügbar sein. Gegebenenfalls muss für das ausgewählte Subnetz eine Ethernet-Karte am Emulations-PC nachgerüstet werden.

Schritt 4: Einstellung der IP-Adresse des Emulationsrechners: Damit eine Verbindung zwischen dem Emulator und dem umgebendem Leitsystem hergestellt werden kann, muss die Ethernet-Schnittstelle konfiguriert werden. Hierfür werden die IP-Adresse, das Subnetz und ggf. das Gateway des Subnetzes eingestellt.

Schritt 5: Ausführen des Emulators auf dem Emulationsrechner: Nachdem die ersten vier Konfigurationsschritte durchgeführt wurden lässt sich der Emulator starten. Eine Instanz wird auf dem Emulations-PC ausgeführt; die in Schritt 2 erzeugte Konfigurationsdatei wird geladen.

Die Konfiguration von weiteren Emulatoren wird analog zu der für FF beschriebenen Vorgehensweise durchgeführt. Für die Emulation eines gesamten Automatisierungssystems, müssen die Emulator-Instanzen auf entsprechend unterschiedliche Rechner verteilt, die benötigte Hardware ggf. nachgerüstet sowie die Hardware und der Emulator selbst konfiguriert werden. Das in diesem Beitrag vorgestellte Framework verfolgt die Automatisierung dieses Vorgehens.

2.2.2 Orchestrierung

Nachdem die Emulation gestartet ist, liest der Emulator die an sein Subsystem gesendeten Eingangswerte ein, führt die Logik der zu emulierenden Komponente aus und schreibt die berechneten Ausgangswerte (vgl. Bild 2) zurück an die übergeordnete AT-Komponente (z.B. das Leitsystem). Die Ein- und Ausgangswerte werden üblicherweise im Engineering-Werk-

zeug der realen Komponente oder der Mensch-Maschine-Schnittstelle des Leitsystems angezeigt und können dort von den Test-Ingenieuren gezielt verändert werden. Zur Kommunikation der Ein- und Ausgangswerte werden die spezifischen Protokolle (FF, Profibus, etc.) der zu emulierenden Komponente verwendet.

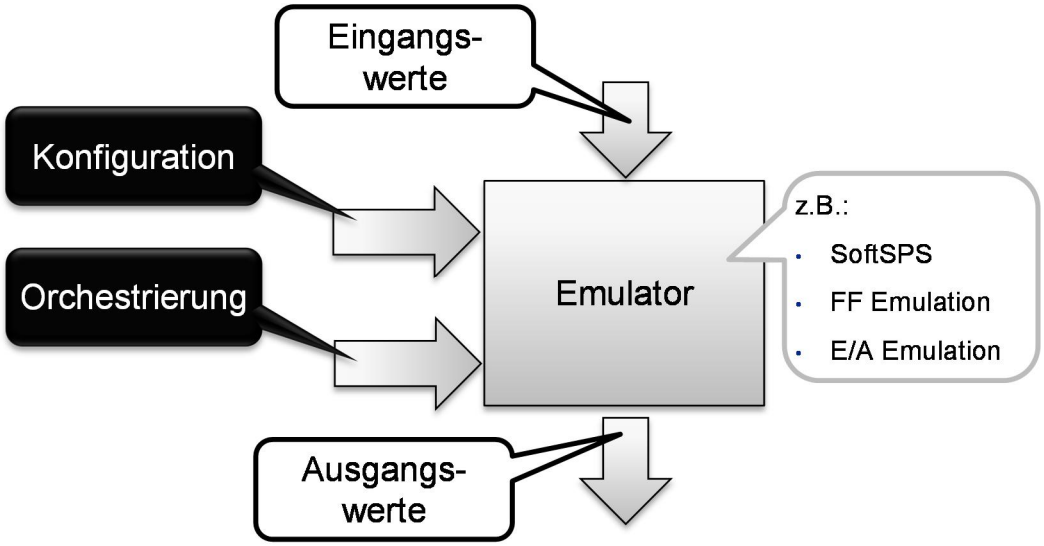

Bild 2: Funktionsweise eines Emulators

Einige Emulatoren stellen zusätzliche Steuerungsfunktionen zur Verfügung. Neben den Basisfunktionen „Start“ und „Stopp“, kann der Nutzer z.B. aktuelle Zustände einfrieren, speichern und diese zu einem späteren Zeitpunkt wiederherstellen.

Ein weiteres Ziel des Frameworks ist es, die notwendige (Fern-) Orchestrierung der Emulatoren zu vereinheitlichen. Hierbei wird das in Anforderung 3 erläuterte einheitliche Bedienkonzept angestrebt, welches sich für jeden Emulator identisch gestaltet. Da die Struktur verschiedener Emulatoren auf ähnlichen Prinzipien beruht liegt es nahe, ein Konzept zur Abstraktion der beschriebenen Funktionalität zu entwerfen. Diese Abstraktion kann als Schnittstelle dienen und sowohl die Konfiguration als auch die zeitgleiche Steuerung mehrerer Emulator-Instanzen ermöglichen.

3. Konzept zur Generierung einer vollständigen Emulation

Die Grundidee des Frameworks besteht darin, verschiedene Emulatoren in einer Umgebung zusammenzufassen. Das Framework muss dabei die Konfiguration und die Steuerung der Emulatoren, unabhängig von deren Typ, auf eine einheitliche Art und Weise durchführen. Zusätzlich soll die Konfiguration der benötigten PC-Hardware automatisch erfolgen. Veränderungen der verwendeten IT-Ressourcen sollen nicht mehr notwendig sein.

Als Basis für das Framework werden moderne Virtualisierungstechnologien gewählt. Der Vorteil dieser Technologien liegt darin begründet, dass diese eine frei konfigurierbare Zusammenstellung des virtualisierten Hardware-Systems (CPU, Ethernetkarte, Speicher, etc.) erlaubt. Auf sich ändernde Hardware-Anforderungen, z.B. durch zusätzlich benötigte Ethernet-Schnittstellen, kann flexibel reagiert werden, ohne reale Komponenten integrieren zu müssen. Die Umgebung ist damit flexibel skalierbar und weitgehend unabhängig von den eigentlichen Emulationsrechnern.

3.1 Automatische Generierung der Emulation

Das Konzept des Frameworks sieht die automatische Generierung von sogenannten Virtuellen Maschinen und darin die automatische Instanziierung von Emulatoren vor. Die Virtuellen Maschinen können im Anschluss an deren Generierung auf unterschiedliche real vorhandene PCs verteilt und darauf ausgeführt werden. Ein im Rahmen des Frameworks entwickeltes Software-Werkzeug übernimmt sowohl die Verteilung als auch die Steuerung der Virtuellen Maschinen bzw. der Emulator-Instanzen.

3.1.1 Voraussetzungen für eine automatische Generierung

Damit eine automatische Generierung der Emulation erfolgen kann wird einmalig, wie in Bild 3 dargestellt, ein Muster einer Virtuellen Maschine aufgesetzt, auf welchem die unterschiedlichen Emulatoren installiert sind. Die Installation ist dabei genauso auszuführen wie auf einem physikalischen PC.

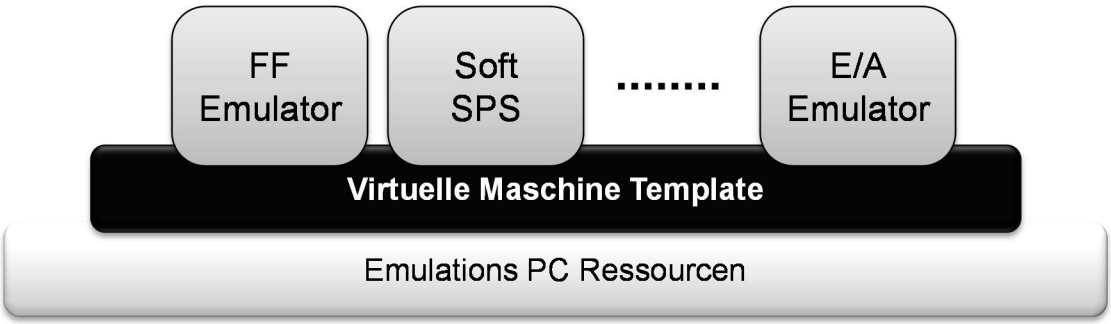

Bild 3: Template einer Virtuellen Maschine mit Emulatoren

Das Muster bietet den Vorteil, dass es vervielfältigt werden kann ohne dafür weitere Hardwareressourcen zu benötigen. Dadurch kann die virtuelle Hardware automatisch verändert und auf die Bedürfnisse der jeweiligen Emulator-Instanz angepasst werden.

Zusätzlich zur Installation auf dem Muster muss der Emulator Konfigurationsparameter für das Framework bereitstellen. Die nachfolgend gelisteten Parameter sind jedoch statisch und werden nicht durch die Konfiguration des Emulators beeinflusst.

1. Nahezu jeder Emulator benötigt eine Ethernet-Schnittstelle. Einige Instanzen benötigen sogar mehrere, da diese z.B. mit einem Client-Server Netzwerk des Leitsystems und einem FF Netzwerk verbunden sind. Aufgrund dessen muss die Anzahl der Ethernet-Schnittstellen für eine Instanz des Emulators parametriert werden.
2. Da Emulatoren instanziiert werden, welche lediglich in einer begrenzten Anzahl auf einem Rechner ausführbar sind, ist die Angabe dieser Begrenzung ebenfalls notwendig.
3. Da einige Emulatoren mehrere Instanzen eines Sub-Systems ausführen können und folglich auch dieser Parameter dem Framework bekannt geben werden muss, ist die Anzahl der Sub-Systeme, welche pro Instanz emuliert werden von Bedeutung.
4. Hinzu kommt ein Parameter für den benötigten Arbeitsspeicher. Wie bei jeder MS Windows Applikation ist es für den Emulator wichtig zu wissen, wie viel Arbeitsspeicher pro Instanz benötigt wird.
5. Letztendlich wird zur Identifikation eines emulierbaren Sub-Systems der zugehörige Systemtyp (z.B. HSE-Subnetz für FF) benötigt. Auf Basis des Typs können die emulierbaren Systeme bestimmt werden.

Speziell bei der Emulation von Feldbussen ist die Anzahl der benötigten IP-Adressen pro Instanz wichtig. Da die IP-Adressen automatisch konfiguriert werden sollen, muss deren Anzahl sowie die Adresswerte selbst bekannt sein. Dieser Parameter ist kein statischer Parameter, sondern variiert mit der zu emulierenden Sub-System-Instanz. Deshalb muss dieser Parameter entsprechend aus den Konfigurationsdaten des Sub-Systems bezogen werden.

Die Werte der erläuterten Konfigurationsparameter sind für jeden Emulator verschieden. Daher müssen diese bei der Integration des Emulators einmalig konfiguriert (mit Ausnahme der IP Adresse) und dem Framework bekannt gegeben werden.

3.1.2 Algorithmus zur Generierung der Emulation

Um die benötigten Virtuellen Maschinen zu erzeugen und die Emulatoren darin zu instanzieren ist ein mehrstufiger Algorithmus anwendbar. Dieser beinhaltet sechs Schritte:

Schritt 1: Exportieren der Topologie des Automatisierungssystems: Zunächst muss die Topologie des Automatisierungssystems aus dem Engineering-System des Leitsystems exportiert werden. Die im Leitsystem modellierte Topologie enthält die automatisierungstechnischen Komponenten, z.B. die verwendeten Steuerungen (Bild 4). Da die emulierbaren

Objekte aufgrund des Objekttyps der Emulatoren identifiziert werden können, kann diese Information dem Framework bereits hier bekannt gegeben werden.

Schritt 2: Auswahl der zu emulierenden Objekte und Export der Konfigurationen: Im zweiten Schritt wird dem Nutzer die Möglichkeit gegeben, die Objekte, die er emulieren möchte, auszuwählen. Während des FAT werden oftmals Teilsysteme emuliert und getestet. Die Auswahl, welcher Teil getestet werden soll, kann in diesem Schritt getroffen werden. Bild 4 zeigt hierzu einen entsprechenden Auswahldialog des Frameworks. Nachdem der Nutzer die Auswahl getroffen hat, werden die Konfigurationsdateien für alle ausgewählten Objekte automatisch erzeugt. Die Konfigurationsdateien werden später benötigt um die Emulator-Instanzen automatisch zu konfigurieren. Bei einer vollständigen Emulation kann die Auswahl einzelner Objekte entfallen.

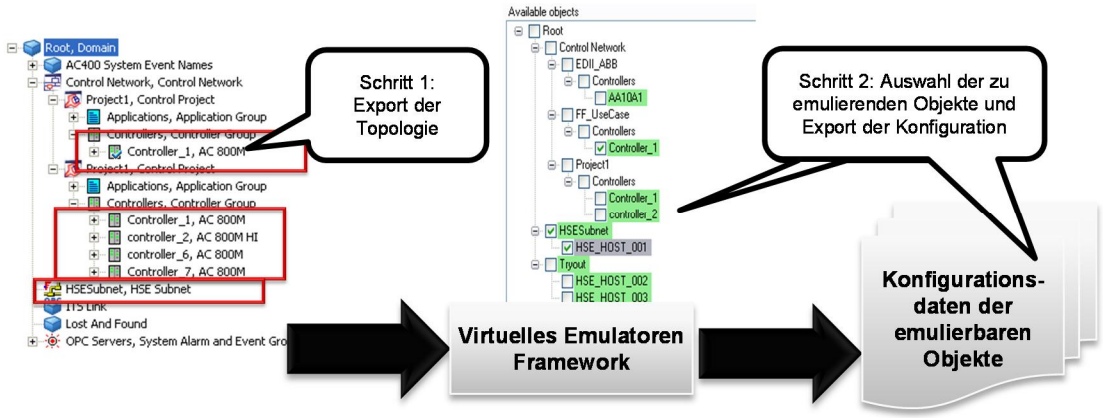

Bild 4: Export der Topologie des Automatisierungssystems und Auswahl der zu emulierenden Objekte

Schritt 3: Bestimmen der Anzahl an Virtuellen Maschinen: Auf Basis der ausgewählten Objekte kann in diesem Schritt die Anzahl der benötigten Virtuellen Maschinen errechnet werden. Hierfür werden folgende Regeln angewandt:

- Die maximale Anzahl ausführbarer Instanzen eines Emulators pro PC darf für keine der Virtuellen Maschinen überschritten werden.
- Die Virtuellen Maschinen haben abhängig von der verwendeten Virtualisierungsumgebung nur eine eingeschränkte Anzahl Ethernet-Schnittstellen zur Verfügung. Diese darf ebenfalls nicht überschritten werden.
- Die Anzahl gleichzeitig emulierbarer Objekte pro Instanz eines Emulators darf für keine Instanz des Emulators überschritten werden.
- Der maximal verfügbare reale Arbeitsspeicher darf für keinen Emulationsrechner überschritten werden. Dies bedeutet, dass die Summe des für die Virtuellen Maschi-

nen eines PCs allokierten Speichers kleiner sein muss als der tatsächliche Speicher des PCs.

Es ist empfehlenswert, stets die minimal notwendige Anzahl Virtueller Maschinen zu verwenden, da mit jeder zusätzlichen VM der Overhead für das Betriebssystem und die Virtualisierungssoftware (auch Hypervisor genannt) steigt. Des Weiteren werden separate Lizenzen für jede Betriebssysteminstanz benötigt.

Schritt 4: Generieren der Virtuellen Maschinen: In diesem Schritt wird die benötigte Anzahl virtueller Maschinen nach dem in Abschnitt 3.1.1 erläuterten Muster erstellt.

Schritt 5: Konfiguration der virtuellen Hardware: Nachdem die Virtuellen Maschinen generiert wurden, muss die virtuelle Hardware entsprechend der Emulatoren, die in den Virtuellen Maschinen ausgeführt werden sollen, konfiguriert werden. Die spezifischen Virtuellen Maschinen enthalten anschließend die benötigte Hardware-Konfiguration (die benötigten Ethernet-Schnittstellen, genügend Arbeitsspeicher) sowie die Emulator-spezifischen IP-Adressen. Sie verhalten sich nach deren Start wie ein Emulationsrechner der auf die Emulatoren abgestimmt wurde.

Schritt 6: Instanziierung der Emulatoren: Nach der Generierung der Virtuellen Maschinen, werden diese auf die Emulationsrechner verteilt.

Die Konfiguration der Hardware ist damit beendet. Die in Schritt 2 erzeugten Konfigurationsdateien können nun für die Konfiguration der erzeugten Emulator-Instanzen angewendet werden.

4. Grenzen des Frameworks und Forschungsbedarf

Das Framework macht sich die Skalierbarkeit einer virtuellen IT-Infrastruktur zunutze. Hierdurch ergeben sich jedoch einige Ausschlusskriterien, die Abschnitt 4.1 erläutert werden. Zusätzlich gibt es weiteren Forschungsbedarf da bisher lediglich die Konfiguration der Virtuellen Maschinen und der Emulatoren betrachtet wurde. Noch ist nicht untersucht, welchen Einfluss die virtuelle Infrastruktur auf das Laufzeitverhalten der Emulation hat.

4.1 Ausschlusskriterien für Emulatoren

Durch den Einsatz einer Virtualisierung können nicht alle Emulatoren in das Framework integriert werden. Folgende Kriterien ergeben sich für die Integrierbarkeit:

- **Das Framework ist ungeeignet für Hardware-basierte Emulatoren**: Emulatoren die als Basis eine Hardwareplattform benutzen (z.B. SIMBA [14]), können nicht in das

Framework integriert werden. Das Framework ist rein auf Software-basierte Werkzeuge ausgelegt.

- **Der Emulator muss auf einer virtuellen Maschine lauffähig sein**: Generell müssen alle Emulatoren - auch Software-basierte - die nicht in einer virtuellen Maschine betrieben werden können von einer Verwendung im Framework ausgeschlossen werden.
- **Ethernet als Kommunikationsbasis**: Der Emulator sollte zur Laufzeit eine Ethernetkommunikation aufweisen. Emulatoren die z.B. über Profibus oder Modbus RTU kommunizieren, benötigen spezielle Hardware, die nicht von der Virtualisierungsumgebung unterstützt wird. Deshalb sind Feldbusse die nicht Ethernet als Hardwareschnittstelle benutzen schwer oder gar nicht integrierbar.

Die Ausschlusskriterien betreffen hauptsächlich ältere Emulatoren. Aktuelle Feldbusse basieren zunehmend auf Ethernet als physikalische Schnittstelle, weshalb davon auszugehen ist, das Ethernet in Zukunft eine wichtige Rolle spielen wird [15]. Emulatoren auf Basis von Hardware sind zwar nicht integrierbar, können jedoch über die konventionellen Kommunikationswege (z.B. Profibus PCI Karte) mit den integrierten Emulatoren kommunizieren und damit im Verbund mit dem Framework verwendet werden.

4.2 Offene Fragen

Der vorliegende Beitrag setzt den Schwerpunkt auf die Konfiguration der Virtuellen Maschinen und der Emulatoren die darin ausgeführt werden. Aus wissenschaftlicher Sicht bleiben bezüglich der Generierung einer vollständigen Emulation folgende Fragestellungen als weiterer Forschungsbedarf offen:

- **Inwiefern wird das Laufzeitverhalten der Emulatoren durch die virtuelle Umgebung beeinflusst?** Um eine belastbare Emulation der Komponenten zu erhalten, müssen die Emulatoren ein mit der realen Komponente identisches Laufzeitverhalten aufweisen. Hierfür müssen Benchmarks entwickelt, angewandt und ausgewertet werden, welche die Emulatoren auf ihre Geschwindigkeit und ihr Echtzeitverhalten untersuchen. Unter anderem ist zu prüfen, ob der PLCopen Benchmark für SPSen verwendet werden kann [16].
- **Wie können vollständige Emulationen geeignet gespeichert werden?** In der Regel wird die Emulation für einen bestimmten Zeitraum benötigt und danach nicht weiter benutzt. Durch eine persistente Speicherung könnte die Emulation zu einem späteren Zeitpunkt weiterverwendet werden (z.B. für Service-Zwecke oder Brown-Field Engineering Projekte).

- **Welche Schnittstellen zur Prozesssimulation werden benötigt?** Um eine vollständige virtuelle Inbetriebnahme zu ermöglichen werden Schnittstellen zwischen der Prozesssimulation und der Emulation benötigt. Diese umfassen sowohl die Orchestrierung als auch den Austausch der Ein- und Ausgangssignale zwischen Emulation und Prozesssimulation [3].
- **Wie könnte ein Konzept zur einheitlichen Benutzerführung aussehen?** Diese Anforderung wurde bisher nur am Rande betrachtet. Ein Konzept zur einheitlichen Benutzerführung und damit dem synchronen Orchestrieren der Emulatoren wird benötigt.
- **Weitere Herausforderungen:** Weitere zu lösende Herausforderungen sind z.B. das Multiuser Management, eine anpassbare Verbindung zum Prozessleitsystem oder SCADA, sowie die Verbindung zwischen verschiedenen Emulationen, dem Engineeringsystem und dem PC, der verwendet wird um die Emulation zu generieren.

5. Fazit

Geographisch verteilte Engineering-Teams, steigende Komplexität und zunehmende Kosten bedingen den Einsatz von Emulation im FAT. Gleichzeitig wird der Einsatz von Emulatoren als Ersatz von Hardwareaufbauten und deren Einsatz zum Test der Logik eines Automatisierungssystems zunehmend komplexer, da jedes Teilsystem, einer jeden Domäne nachgebildet werden muss.

Der vorliegende Beitrag beschreibt eine Methode zur Vereinfachung des Einsatzes von Emulatoren für den ganzheitlichen Test von Automatisierungssystemen. Auf Basis von Virtualisierung wurde ein Konzept zur automatischen Konfiguration von Emulatoren und der benötigten IT-Infrastruktur vorgestellt, welches den Einsatz der Emulatoren und das Entwickeln der Emulationsumgebung für den FAT signifikant vereinfacht.

Das Virtuelle Emulator Framework ist dahingehend eine erweiterbare und dynamische Emulations-Umgebung für die Bereitstellung des gesamten Spektrums an Emulationstechnologien heterogener Domänen, welches sich aufgrund der einheitlichen Nutzerschnittstelle effizient konfigurieren lässt.

Referenzen

[1] Rodies, H.-J.: Planungswerkzeuge aus Sicht des Anlagenbaus. In: Automatisierungstechnische Praxis – atp 44, Heft 1/2002, S. 40-44.

[2] Sato, H: The Recent Movement of Foundation Fieldbus Engineering. SICE Annual Conference in Fukui, 2003, S. 1134-1137.

[3] Barth, M.: Automatisch generierte Simulationsmodelle verfahrenstechnischer Anlagen für den Steuerungstest. Dissertation an der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg, Institut für Automatisierungstechnik, Fortschritt-Berichte VDI, Reihe 20, 2011.

[4] Maeda, T.: A testing environment – ABB's comprehensive suite of software testing and commissioning tools for substation automation systems. ABB Review - IEC61850 Special, 2010, Zürich.

[5] IEC61850: Communication networks and systems in substations, 2003.

[6] Hoernicke, M., Bauer, P.: Emulation dezentraler Steuerungslogik. In: atp-edition, Heft 4/2012.

[7] Hoernicke, M., Weemes, P., Hanking, H.: The fieldbus outside the field – Reducing commissioning effort by simulating Foundation Fieldbus with SoftFF. In: ABB Review, Heft 1/2012.

[8] Wünsch, G.: Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme. Dissertation, Herbert Utz Verlag München, Forschungsberichte IWB Band 215.

[9] Barth, M., Fay, A., Wagner, F., Frey, G.: Effizienter Einsatz Simulations-basierter Tests in der Entwicklung automatisierungstechnischer Systeme. In: Tagungsband "Automation 2010", S. 47-50, 15-16. Juni 2010, Baden-Baden.

[10] Wikipedia: Emulator (http://en.wikipedia.org/w/index.php?title=Emulator&oldid=462149838)

[11] IEC61131-3: Programmable Controllers – Part 3: Programming Languages. Edition 2.0, 2003.

[12] Brandao, D., da Cunha, M.J., Pinotti Jr., M.: Fieldbus Control System Project Support Tool based on Experimental Analysis and Modeling of Communication Bus. IEEE International Conference on Industrial Technology (ICIT), 2004, Volume 2, S. 787-792.

[13] Hoernicke, M., Greifeneder, J.: Next Generation Factory Acceptance Test. In: Jahresbericht 2011 des ABB Forschungszentrums Ladenburg.

[14] SIEMENS Industrial Technologies: SIMBA Profibus – Hardwareplattform, für die Echtzeitsimulation am Profibus im Simulationssystem SIMIT. Broschüre.

[15] Morse, J.: Ethernet Steams Ahead in Asia Pacific. Von IMS Research, Januar 2012. (http://imsresearch.com/news-events/press-template.php?pr_id=2551)

[16] PLCopen: TC3 – Certification – Task Force Benchmarking (http://www.plcopen.org/pages/tc3_certification/benchmarking/)