Modularisierung von Steuerungssoftware

Effizientes Engineering mit IEC 61131 und IEC 61499

M. Melik Merikumians^{ID}, P. Gsellmann^{ID}, G. Schitter^{ID}

Modularisierung und Verteilung der Aufgaben einer Automationsanlage sind die Kernelemente zur Steigerung der Wiederverwendbarkeit von Automationshard- und software. Hierfür werden ein neuer Hierarchisierungsansatz für Automationssysteme, der die Ablaufsteuerung weitestgehend von der Hardwaresteuerung entkoppelt, und eine nahtlose Integration von zyklusbasierten IEC-61131-Systemen in die Entwicklungsmodelle der eventgesteuerten IEC 61499 präsentiert. Beide Ansätze legen hierbei besonderes Augenmerk auf die Wiederverwendbarkeit von Software.

Schlüsselwörter: SPS; Automation; IEC 61131; IEC 61499; Methoden

Modularization of control software: efficient engineering with IEC 61131 and IEC 61499.

Modularization and distribution of the responsibilities of an automation system are important for increasing the reusability of automation hardware and software. For this purpose, a new hierarchization approach for automation systems that decouples the sequence control from the hardware control and a seamless integration of cycle-based IEC 61131 systems into the development models of the event-driven IEC 61499 are presented. Both approaches pay special attention to the reusability of software.

Keywords: PLC; automation; IEC 61131; IEC 61499; methods

Eingegangen am 17. Mai 2021, angenommen am 20. Juli 2021, online publiziert am 19. August 2021
© The Author(s) 2021

1. Einleitung

Das Anlagenengineering wurde in den letzten Jahren immer komplexer. Die Gründe hierfür sind einerseits die immer anspruchsvolleren Produktionsprozesse welche eine Vielzahl verschiedener Technologien miteinander verbinden. Andererseits bringen die geänderten Rahmenbedingungen in den Märkten, welche eine deutlich höhere Variantenvielfalt aufweisen, kürzere Produkt- und Prozesslebenszyklen mit sich [1]. Um die steigende Komplexität beherrschbar zu machen wurde der Fokus auf die Modularisierung der Automationsanlagen gelegt. Die Hardware (HW)-Komponenten wurden hierfür in wiederverwendbare Subgruppen gegliedert und über Feldbusse verbunden. Diese Art der Systemarchitektur wird auch als Cyber-Physical Production System (CPPS) bezeichnet, welche eine Modularisierung und Verteilbarkeit von Automations-HW und Software (SW) fordert. Die Steuerungsprogramme selbst jedoch werden meist weiterhin zentral monolithisch entwickelt und umgesetzt. Der Hauptgrund dafür ist das Abarbeitungsmodell der IEC 61131, der dominante Standard zur Programmierung von Speicherprogrammierbare Steuerungen (SPSen) [2, 3]. Die IEC 61131 nutzt ein zyklisches Abarbeitungsmodell, welches einen eindeutigen Systemzustand während der Programmabarbeitung garantiert. Dieses regelmäßige „Einfrieren“ der realen Welt erleichtert die Programmierung und ist für den Nutzer leicht verständlich. Gleichzeitig ist es jedoch ein Hindernis für die Modularisierung der SW und Aufteilung des Programms über mehrere SPSen, da über mehrere SPSen ein einheitlicher Systemzustand nicht trivial garantiert werden kann. Selbst wenn für alle SPSen die gleiche Zykluszeit gewählt wird, müssen die einzelnen SPS-Zyklen nicht notwendigerweise isochron laufen. Die

SPSen können dadurch unterschiedliche Momentaufnahmen der zu steuernden Anlage haben, was zu Fehlern in der Gesamtapplikation führen kann [4].

Um dieses Problem zu lösen wurde die IEC 61499 entwickelt, welche auch die Modularisierung der SW forciert [5]. Die IEC 61499 nutzt als grundlegendes Programmierelement den Funktionsbaustein (FB), der prinzipiell dem IEC 61131 FB ähnelt, jedoch die datengetriebene Aktivierung der FBs durch eine Eventsteuerung ersetzt. Das bedeutet, dass IEC 61499 FBs nur dann aktiv werden, wenn sie ein Event erhalten haben. Dadurch wird auch das Problem der Synchronisation der Teilsysteme, wie es in IEC 61131 Systemen auftreten kann, vermieden. Dieses Abarbeitungsmodell ist jedoch einerseits ungewohnt für Entwickler die von der IEC 61131 wechseln und andererseits ist die Reihenfolge der FB-Abarbeitung schwieriger nachvollziehbar.

Ein effizientes Anlagenengineering muss daher die Modularisierung und Verteilung ebenso auf der SW-Ebene und gleichzeitig die Weiternutzung des vorhandenen Wissens ermöglichen [6]. Diese Arbeit präsentiert einen Hierarchisierungsansatz in Abschn. 2, welcher zu einer möglichst geringen Abhängigkeit der Module untereinander führt, sowie in Abschn. 3 ein neues Konzept zur Integration von IEC 61131 Systemen in das IEC 61499 Modell.

Melik Merikumians, Martin, Institut für Automatisierungs- und Regelungstechnik (ACIN), TU Wien, Gußhausstraße 27-29/E376, 1040 Wien, Österreich (E-Mail: melik-merikumians@acin.tuwien.ac.at); **Gsellmann, Peter**, Institut für Automatisierungs- und Regelungstechnik (ACIN), TU Wien, Wien, Österreich; **Schitter, Georg**, Institut für Automatisierungs- und Regelungstechnik (ACIN), TU Wien, Wien, Österreich

Abb. 1. In die Widerstandssortieranlage werden Widerstandsplättchen als Schüttgut in den Rüttelförderer eingebracht. Sie werden einzeln und von einem Portalroboter in eine Messstation eingelegt, wo sie mit einem Referenzwiderstand verglichen werden. Basierend auf dem Vergleichsergebnis wird der Widerstand in eine der beiden Ablagepaletten eingelegt

2. Hierarchische Strukturierung von Applikationen

Modularisierung und Hierarchisierung erlaubt die einfache Wiederverwendung von Automationskomponenten in verschiedenen Anlagenteilen und Applikationen. Dazu muss jedoch jede Automationskomponente für sich autark und vollständig sein [7, 8]. Zusätzlich ist es wichtig, dass die Komponenten keinerlei Anforderungen an eine spezifische HW-Implementierung an ihren Schnittstellen weitergeben oder erwarten [8, 9]. Als Beispiel dient die Widerstandssortieranlage in Abb. 1. In die Anlage werden Widerstandsplättchen als Schüttgut in den Rüttelförderer eingebracht. Die Widerstände werden einzeln, der Widerstandswert gemessen und basierend auf dem Messergebnis in eine der beiden Ablagepaletten eingelegt. Folgt man gängigen Methoden zur Hierarchisierung von Automationsystemen [10–12] kann die Anlage in drei Subkomponenten geteilt werden: der Messstation, dem Rüttelförderer und Vereinzelner, und dem Portalroboter mit Vakuumgreifer. Die einzelnen Subkomponenten können nochmals in Subkomponenten aufgeteilt werden. Für die Portalroboter-Komponente wären das die drei Achsen bzw. Achsantriebe für x, y und z, und das Greifwerkzeug. HW-seitig sind diese drei Komponenten vollständig entkoppelt da es keinerlei Abhängigkeiten zwischen den einzelnen Komponenten gibt.

Betrachtet man jedoch die SW-Komponenten so erkennt man, dass die Komponenten aufgrund von Erwartungen an die Prozessparameter miteinander streng gekoppelt sind. Als Beispiel dient die Achsantriebssteuerung einzelner Achsen des Portalroboters (siehe Abb. 2). Der Servomotor-Achsantrieb erwartet als Eingabe die anzufahrende Position in kartesischen Koordinaten (x, y oder z für jede einzelne Achse), sichtbar als Ausgang `Position` des FBs rechts in Abb. 2, was aufgrund des Portalroboteraufbaus nachvollziehbar ist. Diese Annahme wird jedoch an die nächsthöhere Hierarchieebene über die Schnittstelle `Controller_Interface` weitergereicht, die einen konkreten Positionswert (`Final_Position`) erwartet. Die nächsthöhere Komponente wird also gezwungen diese

Abb. 2. Softwarekomponenten eines Servomotors als Achsantrieb. Der grau hinterlegte Teil repräsentiert die Schnittstelle zur nächsthöheren Ebene

Abb. 3. Softwarekomponenten eines pneumatischer Zylinders als Achsantrieb. Der grau hinterlegte Teil repräsentiert die Schnittstelle zur nächsthöheren Ebene

Schnittstelle zu implementieren und bindet sich dadurch stark an Subkomponenten die Koordinaten als Eingabe erwarten.

Das Problem wird evident, wenn man versucht eine Komponente zu verwenden, die eine andere interne Darstellung der Achsposition verwendet, beispielsweise wenn ein pneumatischer Zylinder anstatt des elektrischen Antriebs eingefügt wird. Pneumatische Zylinder werden üblicherweise in die Endanschläge gesteuert da eine genaue Positionsregelung aufwändig ist. Die Fahrweise in die Endanschläge ist für die gewählte Anwendung ausreichend und stellt eine günstigere Variante für den Portalroboter dar. Wenn man nun wieder nach den gängigen Methoden die Modularisierung durchführt, so bekommt man eine SW-Komponente wie in Abb. 3 gezeigt. Wieder sieht man am FB rechts die Ausgänge zur Ansteuerung der HW, welche in diesem Fall den Zylinder entweder einfahren oder ausfahren soll. Links sieht man das neue `Controller_Interface`, welches nun gänzlich anders aussieht als das in Abb. 2 und mit diesem inkompatibel ist. Vervollständigt man das notwendige Automationsprogramm, so sieht man, dass sich diese Schnittstellen bis in die oberste Hierarchieebene ziehen und somit alle SW-Komponente abhängig von den HW-Schnittstellen werden und damit nur beschränkt wiederverwendbar sind.

Um das Problem der Abhängigkeit zur konkreten HW-Implementierung zu lösen müssen die Prozessparameter generalisiert werden. Dies kann dadurch erreicht werden, indem die konkreten Werte, wie sie bisher verwendet wurden, durch symbolische Namen für die Parameterwerte ersetzt werden. Für das konkrete Beispiel der Positionieraufgabe werden Namen für die Positionen über der Entnahmeposition des Rüttelförderers, der Palettenpositionen und der Einlegeposition der Messstation definiert. Dadurch ergibt sich die SW-Komponente für die gesamte Portalrobotereinheit wie in Abb. 4 gezeigt. Die definierten Namen werden von den Steuerungsprogram-

Abb. 4. Die Softwarekomponente der Portalrobotersteuerung (Manipulator). Der grau hinterlegte Teil repräsentiert die Schnittstelle zur nächsthöheren Ebene

Abb. 5. Nach dem MDA-Ansatz werden die HW-spezifischen Codefragmente erst bei der Wahl der endgültigen HW hinzugefügt. Dadurch wird der Quellcode von der HW entkoppelt

men der höheren Hierarchieebenen genutzt um den Programmablauf zu beschreiben. Die symbolischen Positionen werden nun als STRINGS an die unterlagerten HW-Steuerungen über die AAxis-Logic Schnittstelle (siehe Abb. 4) weitergegeben. Diese Schnittstelle beschreibt also nicht wie die der Aktuator vorgefahren werden soll (z. B. Schaltstellungen oder kartesische Koordinaten), sondern nur wohin er vorgefahren werden soll.

Die Namen sind für die konkrete HW-Steuerungskomponente bedeutungslos da entweder Koordinaten oder Zylinderstellungen erwartet werden. Die Namen werden in einem Model Driven Architecture (MDA)-Ansatz entsprechend zur aktuell verwendeten HW-Komponente gemäß einer vorab definierten Tabelle übersetzt (siehe Abb. 5).

Durch diese Methode können die überlagerten Komponenten, welche meist nur allgemeine Abläufe modellieren, von den spezifischen HW-Komponenten entkoppelt werden und somit die Wiederverwendbarkeit der beteiligten HW- und SW-Komponenten erhöhen.

3. Integration von IEC 61131-3 in IEC 61499

Die bisherigen Ansätze einer kombinierten Laufzeitumgebung für IEC 61131 und IEC 61499, mit dem Ziel jegliche Steuerungslogik zu unterstützen, können grob in drei Klassen eingeteilt werden [2]:

- Getrennte IEC 61131 und IEC 61499 Laufzeitumgebungen verbunden über Kommunikationsschnittstellen.
- Erweiterte IEC 61131 Laufzeitumgebung mit der Fähigkeit eventbasierte IEC 61499 Logik auszuführen.
- Erweiterte IEC 61499 Laufzeitumgebung mit der Fähigkeit zyklische IEC 61131 Logik auszuführen.

Die erste Klasse beschreibt ein lose gekoppeltes System, in dem zwei separate Laufzeitumgebungen implementiert sind, welche über eine Kommunikationsschnittstelle interagieren. Dies ermöglicht die Ausführung von IEC 61131-3 und IEC 61499 Applikationen in einem Gerät, erhöht jedoch den Entwicklungsaufwand und unterstützt keine Wiederverwendung von Elementen zwischen den Standards.

Ein Vertreter der zweiten Klasse stellt ISaGRAF [13] von Rockwell Automation dar. Hierbei werden in einer RESOURCE IEC 61499 FB in einer vorbestimmten Reihenfolge zyklisch aufgerufen. Während der Aktivierung können eintreffende Events verarbeitet werden. Obwohl mit dieser Implementierung IEC 61131-3 und IEC 61499 Applikationen nebeneinander verwendet werden können, sind einige Nachteile, wie der Abarbeitungs-Overhead durch das zyklische Aufrufen, die Berechnung der Abarbeitungsreihenfolge, sowie nicht konforme Implementierung von IEC 61499 Konzepten gegenwärtig [14].

Das dritte Konzept erscheint als das flexibelste und damit vorteilhafteste der vorgestellten Konzepte [2]. Neben der Wiederverwendung von existierendem IEC 61131-3 Code, erlaubt diese Methode auch die enge Kopplung zu IEC 61499 Steuerungslogik und ermöglicht die Verteilung auf verschiedenen Ressourcen. Eine bereits bestehende Implementierung liegt bereits von nxtControl [15] vor. Ein Dienstschnittstellen-FB koppelt den zyklischen Teil mit dem eventgetriebenen Teil der Anwendung. Jedoch führt der Kopplungs-FB zum Entstehen gravierender Nachteile, wie instabile Zykluszeiten und Dateninkonsistenzen [16].

3.1 Integrationskonzept

Wie die Analyse der verschiedenen Implementierungsoptionen zeigt, stellt die IEC 61499 die bessere Wahl als Basis für die Erstellung eines gemeinsamen Systemmodellierungskonzeptes dar. Zum einen bietet die IEC 61499 native Unterstützung von Verteilung der Steuerungslogik, zum anderen erscheint ihr das Abarbeitungsmodell besser geeignet zur Beherbergung einer IEC 61131 Laufzeitumgebung. Diese Argumente motivieren die Entwicklung eines neuen kombinierten Ansatzes basierend auf der IEC 61499, welcher schematisch in Abb. 6 dargestellt ist.

3.1.1 Zuordnung der Modellelemente

Basierend auf einer Analyse der Korrespondenzen zwischen IEC 61131-3 und IEC 61499 [6], erscheint die abarbeitungsorientierte Zuordnung als die naheliegendere Wahl hinsichtlich des Integrationskonzeptes.

Da ein IEC 61131-3 TASK mit einer IEC 61499 RESOURCE korrespondiert, wird ein spezieller Ressourcentyp erstellt, der als Abarbeitungscontainer für IEC 61131-3 Programm-Organisationseinheiten (POEs) dient und die beinhalteten IEC 61131 POEs zyklisch aufruft. Die geforderte Zykluszeit wird dieser speziellen die RESOURCE als Parameter bekannt gemacht. Die IEC 61499 definiert die RESOURCE als eine von anderen RESOURCES unabhängige Abarbeitungseinheit. Unter der Annahme, dass die Echtzeitbedingungen für alle Ressourcen erfüllt werden können, kann eine korrekte zyklische Abarbeitung garantiert werden. Dies kann beispielsweise dadurch erreicht werden, dass jede RESOURCE als Echtzeit-Thread mit Round-Robin Scheduling gestartet wird. Um die zyklische Abarbeitung von IEC 61131 POEs in das eventbasierte Abarbeitungsmodell der IEC 61499 zu integrieren, wird die Aufrufreihenfolge als lineare Liste in die Event-Aufrufliste eingetragen (siehe Abb. 7). Im Gegensatz zu IEC 61499 Funktionsbausteinnetzwerken, deren Abarbeitungsreihenfolge dynamisch erzeugt werden muss, kann die Aufrufreihenfolge einer Funktionsbaustein-Sprache (FBS) Applikation statisch berechnet werden (siehe Abschn. 3.1.2). Bei jedem zyklischen Aufruf der IEC 61131 RESOURCE wird die Abarbeitung am Anfang der

Abb. 6. Schematische Repräsentation des Integrationsansatzes: Die vorgeschlagene Entwicklungsumgebung unterstützt das Engineering von IEC 61131-3 und IEC 61499 Applikationen. Darüber hinaus kann der EFB bzw. der IEC-61131 FB als gemeinsames SW-Element verwendet werden. Die adaptierte IEC 61499 Laufzeitumgebung ermöglicht die Verwendung von IEC 61499 EMB_RES Ressourcen, und in IEC 61499 Ressourcen eingebettete IEC 61131-3 TASKS. Alle Aspekte des IEC 61499 Gerätemodells [17] können von beiden Ressourcentypen verwendet werden

(a) Eventabarbeitung: Auftretende Events werden zu der Liste hinzugefügt und nacheinander abgearbeitet. Ein Eventeintrag wird nach der Ausführung entfernt. Der START Zeiger gibt an, welches Event als nächstes ausgeführt wird, während der END Zeiger angibt, an welcher Stelle das nächstauftretende Event hinzugefügt wird.

(b) Adaption der Eventabarbeitung für das zyklische IEC 61131-3 Abarbeitungsmodell: Function Block (FB)e (eigentlich deren REQ Events) werden in der statisch berechneten Reihenfolge (siehe Algorithmus 1) hinzugefügt, aber im Gegensatz zu Abb. 7a, nie entfernt. Nach dem Ausführen des letzten FBs stoppt die Eventabarbeitung bis zum nächsten Zyklus und stellt somit eine konstante Zykluszeit her.

Abb. 7. Abarbeitungsprinzip der (a) IEC 61499 Eventliste und (b) des IEC 61131-3 Abarbeitungsmodells basierend auf der Eventliste

Liste gestartet und endet mit der Abarbeitung des letzten einge-tragenen FB. Danach wird bis zum nächsten Aufruf des IEC 61131 RESOURCE zum Start des nächsten Zyklus gewartet. Hiermit ist eine parallele Abarbeitung von zyklischer IEC 61131-3 und eventgetriebener IEC 61499 Steuerungslogik möglich.

Ein weiterer wichtiger Punkt einer kombinierten Programmier-umgebung ist Code-Sharing zwischen den beiden Welten um Wartungs- und Entwicklungsaufwand zu reduzieren. Der einfache FB (EFB) erweist sich als geeignetes eventgetriebenes Ana-logon zum IEC 61131-3 FB. Daher wird liegt in dieser Arbeit der Fokus auf der Integration der IEC 61131-3 FBs. Die IEC 61499 be-schreibt den EFB als einen Dienstschnittstellen-FB, bei dem jedes verfügbare Eingangs-event jeweils die Abarbeitung eines asso-zierten Algorithmus startet. Für das vorliegende Integrationskon-zept [16] wird diese Definition adaptiert: Ein EFB gilt als vereinfach-te Version eines Basis-FB mit nur einem Algorithmus, nur einem Ein-gangsevent REQ, und einem Ausgangsevent CNF. Wird nun schluss-endlich der Eventkopf des IEC 61499 FB entfernt, so kann ein IEC 61499 EFB auch als IEC 61131-3 FB verwendet werden. Ein FB der abgearbeitet werden soll, wird durch das REQ Event des dualen EFB aktiviert (siehe Abb. 8). Damit werden die Datenwerte, welche an den Eingängen des FB anliegen, übernommen und die Abarbeitung des FB gestartet. Das CNF Event wird in der Abarbeitung nur zur Be-

Abb. 8. Der adaptierte EFB kann als IEC 61131-3 FB verwendet wer-den, womit der Entwicklungsaufwand für ein gemeinsames System reduziert wird. Hier wird als Beispiel der F_AND FB in seiner dualen Repräsentation gezeigt

stimmung des Endes der Abarbeitung und der damit einhergehen-den Aktualisierung der Ausgänge verwendet. Algorithmen, welche Sprachen verwenden deren Abarbeitungsreihenfolge im zyklischen Abarbeitungsmodell der IEC 61131 inhärent wohldefiniert sind (z. B. Structured Text (ST), Kontaktplan (KOP), FBS), können trivial über-nommen werden. Algorithmen die als IEC 61499 FB-Netzwerk (FBN) definiert sind, müssen gesondert betrachtet werden. Sie ähneln in ihren Eigenschaften den zusammengesetzten FB der IEC 61499. Diese können als atomare Abarbeitungseinheiten gesehen werden [18], was es erlaubt jeden dieser FBN Algorithmen gesondert zu betrachten. Eine Möglichkeit eventbasierte FBNs in das zyklische Abarbeitungsmodell der IEC 61131 zu übertragen wurde von Las-tra et al. [19] präsentiert. Eine Einschränkung des Code-Sharing ist, dass VAR_IN_OUT, ACCESS_PATH und GLOBAL Variablen nicht genutzt werden können, da diese im Sprachgebrauch der IEC 61499 nicht existieren. FBs die in der IEC 61499 über ein INIT Event ini-tialisiert werden müssen, können ebenfalls verwendet werden. Die INIT Events werden beim Starten der RESOURCE vor dem ersten Zyklus aktiviert und die entsprechenden Initialisierungsalgorithmen ausgeführt. Herstellerspezifische- und Kommunikations- FBs können ebenfalls über die Abarbeitungsmodelle geteilt werden, solange die oben genannten Einschränkungen nicht verletzt werden.

Mit dem vorgestellten Zuordnungskonzept der SW-Elemente ist eine Koexistenz von IEC 61499 und IEC 61131-3 bereits möglich, jedoch noch keine Interaktion. Da die IEC 61499 die Datentypen der IEC 61131-3 direkt übernommen hat kann der Datenaustausch mit den IEC 61499 Kommunikations-FBs ermöglicht werden. Hier-bei sind zwei Kommunikationsmodelle präsent: Unidirektionale In-teraktion mit den PUBLISH/SUBSCRIBE FBs und bidirektionale In-teraktion mit den CLIENT/SERVER FBs. Diese FBs werden in die IEC 61131-3 übertragen und ermöglichen so Code-Sharing und Interak-tion zwischen den beiden Subsystemen.

3.1.2 Abarbeitungsverhalten

Schlussendlich wird das Abarbeitungsverhalten von IEC 61131-3 FBS Programmen in dem kombinierten Laufzeitsystem definiert. Dieses Verhalten soll dem in der IEC 61131-3 definierten Abarbeitungsverhalten entsprechen. Dieses ist wie folgt definiert [20, Kapitel 8.1.5.1]:

1. Kein Element eines Netzwerks darf ausgewertet werden, bevor die Zustände all seiner Eingänge bekannt sind.
2. Die Auswertung eines Netzwerkelements ist erst dann abgeschlossen, wenn alle Ausgänge des Elements ermittelt wurden.
3. Die Auswertung eines Netzwerks ist erst dann abgeschlossen, wenn alle Ausgänge aller Netzwerkelemente ermittelt wurden.

Für die weitere Analyse wird diese informelle Beschreibung in ein formales Modell überführt. Ein FBs wird hierfür als 2-Tupel $F = (I, O)$ beschrieben, mit I als FB-Eingänge und als O die FB-Ausgänge. In dieser Darstellung werden Eingänge der POE welche das FBS beinhaltet als $(\emptyset, O_m)$ und POE-Ausgang als $(I_n, \emptyset)$ dargestellt. POEs können in diesem Zusammenhang IEC 61131-3 PROGRAMS, FBS, oder FUNCTIONS sein. Das Netzwerk lässt sich als 4-Tupel $P = (\Sigma_I, \Sigma_O, \Sigma_F, C)$ beschrieben werden, wobei

- Σ_I die Menge aller FB-Eingänge und POE-Ausgänge I_n ,
- Σ_O die Menge aller FB-Ausgänge und POE-Eingänge O_m ,
- Σ_F die Menge aller FBs F_i ,
- C die $|\Sigma_I| \times |\Sigma_O|$ ungewichtete Verbindungsmatrix aller Verbindungen zwischen den Elementen aus Σ_I und Σ_O

darstellt. Die Berechnung der statischen Abarbeitungsreihenfolge einer in FBS geschriebenen POE lässt sich als Suchproblem auf dem 4-Tupel P (vgl. Algorithmus 1) formulieren. Im Falle von impliziten oder expliziten Rückkopplungen im FBS Netzwerk müssen Initialwerte für Rückkopplungspfade angegeben werden, um die Berechnung einer Abarbeitungsreihenfolge zu ermöglichen [20, Kapitel 8.1.5.2] [21]. Dadurch wird das zirkuläre Problem des Rückkoppelpfads aufgelöst und ermöglicht die Berechnung einer korrekten, jedoch nicht unbedingt eindeutigen, Abarbeitungsreihenfolge in allen Fällen. Als Beispiel hierfür dient das FBS Netzwerk in Abb. 9. Die Eingänge IN1, IN2 und IN3 sind am Anfang des Programmablaufs durch das Prozessabbild der Eingänge (PAE) bekannt. Da IN1 und IN2 die Zustände der Eingänge von FB1 definieren, kann FB1 ausgewertet werden. Ähnliches gilt für FB2, dessen einziger Eingang durch IN2 eindeutig definiert wird und deshalb ebenfalls abgearbeitet werden kann. Der Eingang IN3 führt zu FB3 und definiert damit den Zustand von Eingang 1 von FB3. Eingang 2 von FB3 ist in einem expliziten Rückkopplungspfad mit Ausgang 3 von FB3. Ohne Initialwert könnte FB3 daher nie ausgeführt werden, da die Ausgänge von FB3 nur ermittelt werden, wenn alle Eingänge wohldefiniert sind, was jedoch ohne den Wert von Ausgang 3 nicht möglich ist. Durch die Angabe eines Initialwerts an Eingang 2 sind jedoch alle Eingänge von FB3 wohldefiniert und FB3 kann ausgewertet werden. Das aktuelle Ergebnis von Ausgang 3 wird im nächsten Abarbeitungszyklus als Wert für Eingang 2 wieder herangezogen. Nun da alle FBs die direkt mit Eingängen verbunden sind ausgewertet wurden, wird ermittelt welche nachfolgenden FBs ausgewertet werden können. Die Ausgänge von FB1 und FB2 sind nun bekannt und definieren die Eingänge von FB4 welcher nun ebenfalls ausgewertet werden kann, womit nun auch alle Eingänge von FB5 über die Ausgänge von FB3 und FB4 bekannt sind, und damit ausgewertet werden kann. Damit sind nun die Ausgänge der POE bekannt und das FBS Netzwerk vollkommen ausgewertet. Wie eingangs erwähnt gibt es im Allgemeinen keine eindeutige Abarbeitungsreihenfolge. Die Abarbeitung von FB1, FB2 und FB3 kann beliebig vertauscht werden. Auch FB4 kann vor FB3 ausgewertet werden, solange die Auswertung von FB1 und FB2 vor FB4 passiert.

Abb. 9. Statische Berechnung der FBS Abarbeitungsreihenfolge gemäß der IEC 61131-3 [22]. In diesem Beispiel wird die ermittelte Abarbeitungsreihenfolge des Netzwerks durch römische Zahlen dargestellt. Bei Rückkopplungen muss ein Initialwert vorgegeben werden

Algorithmus 1 Berechnung der Abarbeitungsreihenfolge

Eingang: Liste der Eingänge I , Liste der Ausgänge O , Liste der FBs F , Liste der Verbindungen C

Ausgang: Abarbeitungsreihenfolge S

```

1: Entferne alle vorinitialisierten Eingänge von  $I$ 
2: while  $F$  nicht leer do
3:   if Ein Element von  $F_i$  0 Eingänge hat then
4:      $S \leftarrow F_i$ 
5:   Entferne alle Ausgänge  $O_i$  von  $F_i$ 
6:   Entferne alle über  $C$  verbundenen Eingänge  $I_i$ 
7:   Entferne  $F_i$  von  $F$ 
8:   end if
9: end while

```

3.2 Zusammenfassung der Integration von IEC 61131-3 in IEC 61499

Mit dem vorgestellten Integrationskonzept [16] ist eine gemeinsame Entwicklungs- und Laufzeitumgebung für die kombinierte Verwendung von IEC 61131-3 und IEC 61499 geschaffen. Die beschriebene Zuordnung der SW-Elemente ermöglicht die Erstellung eines Steuerungsgerätes, das sowohl die Integration von bestehenden IEC 61131-3 Systemen als auch die Verteilungsaspekte der IEC 61499 unterstützt. Darüber hinaus kann der BFB als gemeinsames SW-Element von beiden Subsystemen verwendet werden, und so den Entwicklungs- und Ausführungsaufwand reduziert.

4. Zusammenfassung

Dieser Artikel präsentiert neuartige Ansätze zur Hierarchisierung von Automatisierungssystemen und zur Integration von IEC 61131 Applikationen in IEC 61499 Systeme. Der vorgestellte Hierarchisierungsansatz erlaubt es die Ablaufsteuerung von der HW-Ansteuerung weitestgehend zu entkoppeln, was eine effektive Wiederverwendung von Automations-SW ermöglicht. Der Integrationsansatz ermöglicht es zyklische Programme nach IEC 61131 in die IEC 61499 Systemarchitektur einzugliedern und dadurch die zyklischen Teilsysteme über den Eventmechanismus der IEC 61499 voneinander zu entkoppeln. So können verteilte IEC 61131 Anwendungen einfach realisiert und das vorhandene Expertenwissen auch in der IEC 61499 weiter genutzt werden. Beide Methoden verringern den Engineeringaufwand für zukünftige Automatisierungsprojekte durch die verbesserte Wiederverwendbarkeit und Modularisierung.

Funding Note Open access funding provided by TU Wien (TUW).

Hinweis des Verlags Der Verlag bleibt in Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutsadressen neutral.

Open Access Dieser Artikel wird unter der Creative Commons Namensnennung 4.0 International Lizenz veröffentlicht, welche die Nutzung, Vervielfältigung, Bearbeitung, Verbreitung und Wiedergabe in jeglichem Medium und Format erlaubt, sofern Sie den/die ursprünglichen Autor(en) und die Quelle ordnungsgemäß nennen, einen Link zur Creative Commons Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Die in diesem Artikel enthaltenen Bilder und sonstiges Drittmaterial unterliegen ebenfalls der genannten Creative Commons Lizenz, sofern sich aus der Abbildungslegende nichts anderes ergibt. Sofern das betreffende Material nicht unter der genannten Creative Commons Lizenz steht und die betreffende Handlung nicht nach gesetzlichen Vorschriften erlaubt ist, ist für die oben aufgeführten Weiterverwendungen des Materials die Einwilligung des jeweiligen Rechteinhabers einzuholen. Weitere Details zur Lizenz entnehmen Sie bitte der Lizenzinformation auf <http://creativecommons.org/licenses/by/4.0/deed.de>.

Literatur

- Koren, Y. (2010): The global manufacturing revolution: product-process-business integration and reconfigurable systems. Wiley series in systems engineering and management. New York: Wiley.
- Sünder, C., Zötl, A., Christensen, J. H., Steininger, H., Ritsche, J. (2008): In 2008 6th IEEE international conference on industrial informatics. New York: IEEE. <https://doi.org/10.1109/indin.2008.4618109>.
- Salazar, L. A. C., Alvarado, O. A. R. (2014): In 2014 III international congress of engineering mechatronics and automation (CIIMA) (S. 1–5). <https://doi.org/10.1109/CIIMA.2014.6983434>.
- Schoop, R., Strelzof, A. (1996): Control Eng. Pract., 4(6), 855. [https://doi.org/10.1016/0967-0661\(96\)00078-0](https://doi.org/10.1016/0967-0661(96)00078-0).
- Hegny, I., Strasser, T., Melik-Merkumians, M., Wenger, M., Zötl, A. (2012): In Proceedings of 2012 IEEE 17th international conference on emerging technologies and factory automation (ETFA 2012). <https://doi.org/10.1109/etfa.2012.6489639>.
- Sunder, C., Wenger, M., Hanni, C., Gosetti, I., Steininger, H., Ritsche, J. (2008): In 2008 IEEE international conference on emerging technologies and factory automation (S. 369–376). <https://doi.org/10.1109/ETFA.2008.4638420>. ISSN: 1946-0759.
- Hametner, R., Zötl, A., Semo, M. (2010): In 2010 IEEE international conference on automation science and engineering (S. 156–161). <https://doi.org/10.1109/COASE.2010.5584013>.
- Melik-Merkumians, M., Wenger, M., Hametner, R., Zötl, A. (2010): In Proceedings IEEE emerging technologies and factory automation (ETFA 2010). <https://doi.org/10.1109/ETFA.2010.5641252>.
- Wenger, M., Melik-Merkumians, M., Hegny, I., Hametner, R., Zötl, A. (2011): In 2011 IEEE conference on automation, science and engineering, August 24–27, 2011, Trieste, Italy, proceedings (S. 495–500). <https://doi.org/10.1109/case.2011.6042458>.
- Zötl, A., Prähofer, H. (2013): IEEE Trans. Ind. Inform., 9(4), 2387. <https://doi.org/10.1109/tii.2012.2235449>.
- Faldella, E., Paoli, A., Tili, A., Sartini, M., Guidi, D. (2009): In 2009 XXII international symposium on information, communication and automation technologies (S. 1–7). Sarajevo: IEEE. <https://doi.org/10.1109/ICAT.2009.5348451>. <http://ieeexplore.ieee.org/document/5348451/>.
- Tilli, A., Paoli, A., Sartini, M., Bonivento, C., Guidi, D. (2009): In 2009 IEEE conference on emerging technologies & factory automation (S. 1–8). Palma de Mallorca: IEEE. <http://ieeexplore.ieee.org/document/5347063/>. <https://doi.org/10.1109/ETFA.2009.5347063>.
- Technology, I. (2021). https://www.rockwellautomation.com/en-id/support/documentation/technical-data/isagrar_20190326-0743.html.
- Thramboulidis, K. (2012): In Proceedings of 2012 IEEE 17th international conference on emerging technologies factory automation (ETFA 2012) (S. 1–8). <https://doi.org/10.1109/ETFA.2012.6489672>. ISSN: 1946-0759.
- Mayer, H. (2013): Control & drives. <http://www.computer-automation.de/steuerungsebene/steuern-regeln/artikel/93521/>.
- Gsellmann, P., Melik-Merkumians, M., Zötl, A., Schitter, G. (2021): IEEE Trans. Ind. Inform., 17(8), 5411–5418. <https://doi.org/10.1109/TII.2020.3033330>. <https://ieeexplore.ieee.org/document/9238463/>.
- IEC TC65/WG6 (2005): IEC 61499: Function blocks for industrial-process measurement and control systems – Parts 1 to 4. Geneva: International Electrotechnical Commission (IEC).
- Strasser, T., Zötl, A., Christensen, J. H., Sünder, C. (2011): IEEE Trans. Syst. Man Cybern., Part C, Appl. Rev., 41(1), 41. <https://doi.org/10.1109/TSMCC.2010.2067210>.
- Lastra, J., Godinho, L., Lobov, A., Tuokko, R. (2005): In INDIN '05. 2005 3rd IEEE international conference on industrial informatics, 2005 (S. 80–85). <https://doi.org/10.1109/INDIN.2005.1560356>. ISSN: 2378-363X.
- IEC TC65/WG6 (2003): IEC 61131: Standard - Programmable controllers – Parts 1 to 8. International Electrical Commission.
- Lewis, R. W. (1998): Programming industrial control systems using IEC 1131-3. Institution of Engineering and Technology. https://www.ebook.de/de/product/23703035/r_w_lewis_programming_industrial_control_systems_using_iec_1131_3.html
- International Electrotechnical Commission (2013): IEC 61131 – Programmable controllers, Part 3: Programming languages.

Autoren

Martin Melik Merkmumians

Martin Melik Merkmumians absolvierte 2009 sein Masterstudium Automatisierungs- und Regelungstechnik an der TU Wien und begann unmittelbar danach ein Doktoratsstudium am Institut für Automatisierungs- und Regelungstechnik (ACIN) der TU Wien in der Gruppe für Advanced Mechatronic Systems. Seine Forschungsthemen sind komponentenbasierte Automatisierung, serviceorientierte

Methoden und selbstlernende Algorithmen in Automatisierungssystemen, sowie wissensbasierte Entwicklungsmethoden für industrielle Automatisierungssoftware.

Peter Gsellmann

Peter Gsellmann absolvierte sein Masterstudium Energie- und Automatisierungstechnik an der Technischen Universität Wien (2018) und ist aktuell als Doktorand am Institut für Automatisierungs- und Regelungstechnik (ACIN) der TU Wien tätig. Seine primären Forschungsthemen umfassen industrielle Automation, sowie kamerageführte Steuerung von Industrierobotern.

Georg Schitter

Georg Schitter ist Professor für Advanced Mechatronic Systems am Institut für Automatisierungs- und Regelungstechnik (ACIN) der TU Wien. Er absolvierte das Studium Elektrotechnik an der TU Graz, Österreich (2000) sowie ein Diplom in Informationstechnik und das Doktoratsstudium an der ETH Zürich, Schweiz (2004). Nach mehrjährigen Forschungs-

aufenthalten an der UCSB, USA, und TU Delft, Niederlande wurde er 2010 an die TU Wien berufen. Seine primären Forschungsinteressen befassen sich mit der Entwicklung integrierter Systemlösungen mechatronischer Systeme, inklusive modularer Automation, neuer system- und regelungstechnischer Methoden, vor allem für Anwendungen in der Hochtechnologie, wissenschaftlicher Instrumentierung und bildgebende mechatronische Systeme, wie AFM, scanning Laser und LIDAR-Systeme, Teleskopsysteme, Adaptive Optik, und Lithographiesysteme für die Halbleiterindustrie.