GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Abschlussbericht

IMPACT

Innovative Methods for Programming of Automation Control Technology

Hochschule Pforzheim, Institut für Smart Systems und Services, Tiefenbronner Str. 65, 75175 Pforzheim

Dokument-Version: 1.0
Datum: 31.03.2023
Berichtszeitraum: 01.10.2019 – 30.09.2022
Verbreitungsgrad: Öffentlich
Projekt: Impact
Förderkennzeichen: 01IS19031B
Laufzeit: 01.10.2019 – 30.9.2022

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS190318 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Autoren

Prof. Dr.-Ing. Thomas Greiner, Hochschule Pforzheim

Dr. rer. nat. Grischan Engel, Hochschule Pforzheim

Christoph Lehnert M. Sc., Hochschule Pforzheim

Inhalt

1 Ergebnisse

1.1 Aufgabenstellung

Die Hochschule Pforzheim arbeitete sowohl an der Konzeption und Umsetzung der domänenspezifischen Sprache (DSL) als auch an der Entwicklung eines Interpreters zur Ausführung der DSL. Weiterhin erfolgte die Evaluation durch Integration der erarbeiteten Lösungen in einen bereits vorhandenen I4.0-Demonstrator.

1.2 Projekt- und Qualitätsmanagement

Im Projektzeitraum wurden zur Koordination und engen Abstimmung der Projektpartner im zweiwöchigen Rhythmus Videokonferenzen abgehalten. Weiterhin wurde ein Content-Management-System zum Austausch von Projektinformationen genutzt. Im November 2019 und im Mai 2022 fanden eintägige Treffen der Projektpartner statt (20.11.2019 Fraunhofer IESE, Kaiserslautern, 5.5.2023 Hochschule Pforzheim). Aufgrund der Covid-19 Pandemie waren keine weiteren Treffen in Präsenz möglich.

1.3 Wissenschaftlich-technische Ergebnisse

Zur Softwareentwicklung für Produktionsprozesse wurde eine hierarchische und serviceorientierte DSL entworfen und ausgearbeitet. Neben Logik- und Datenstrukturen bietet diese DSL je nach Anwendungsdomäne (Prozesstechnik oder Fertigungstechnik), passende Sprachkonstrukte. Weiterhin werden spezialisierte Modellierungselemente zur direkten Einbindung von Verwaltungsschalen und zur Ausführung von verteilten Automatisierungsdiensten bereitgestellt. Hierzu werden spezifische Diensteschnittstellen und Kommunikationsprotokolle genutzt und vorhandene System- und Prozessobjekte berücksichtigt. Auf Grundlage der beschriebenen DSL wird die automatisierte Erzeugung von interpretierbarem Code auf Grundlage des Software-Frameworks Eclipse Xtext ermöglicht. Daraufhin kann der generierte Code mittels einer Hardwareabstraktionsschicht, welche standardisierte Schnittstellen zur Interaktion mit Aktoren und Sensoren bereitstellt, auf der Zielplattform ausgeführt werden. Abschließend wurde die Evaluation der DSL und des Interpreters an der I4.0 Modellanlage des Instituts für Smart Systems und Services an der HS Pforzheim durchgeführt.

1.3.1 Domänenspezifische Sprache

Im Rahmen des Vorhabens wurden mögliche DSL-Konzepte sowie entsprechende Softwareframeworks zur Implementierung auf Vor- und Nachteile hin analysiert. Darauf basierend wurde das Konzept für eine hierarchische und serviceorientierte DSL entwickelt und implementiert.

1.3.2 Konzept für eine hierarchische DSL

Das grundlegende Konzept der DSL weist die folgenden Merkmale auf:

- Die Komplexität von Prozessablaufbeschreibungen wird mittels hierarchischen Abstraktionsebenen reduziert.
- Damit wird auch eine kooperative Zusammenarbeit zwischen Automatisierungsingenieuren und Informatikern bei der Erstellung von Automatisierungssoftware erleichtert.
- Es wird die Nutzung von Informationen aus Verwaltungsschalen ermöglicht, welche einen einfachen Zugriff auf Prozessparameter erlaubt.

Das Konzept der serviceorientierten, hierarchisch strukturierten DSL mit Anbindung an Verwaltungsschalen wird in Abbildung 1 veranschaulicht. Diese besteht aus vier aufeinander aufbauenden Ebenen, die eine granulare Verfeinerung bzw. Konkretisierung des Programmablaufs zulassen. Die Ebenen sind strikt hierarchisch organisiert, d.h. es können keine Ebenen übersprungen werden. Weiterhin unterscheiden sich die Ebenen in der bereitgestellten Funktionalität und der Abstraktionsgrad der Ebenen nimmt nach unten hin ab.

Ebene 4 (Process Service Workflow) bietet den höchsten Abstraktionsgrad und dient zur Definition des grundlegenden Prozessablaufs. Dieser wird von einem Prozessingenieur mittels einzelnen aufeinanderfolgenden Prozessoperationen bzw. Fertigungsschritten, abgeleitet von einem Rezept oder einem Ablaufplan, erstellt. Dazu nutzt er von Softwarearchitekten entwickelte Funktionsbausteine, welche auf Ebene 3 (Application Service Interfaces) bereitgestellt werden. Diese Funktionsbausteine bilden die Schnittstelle zu Steuerungen/Regelungen für standardisierte Prozessabläufe. Die konkrete Implementierung dieser Abläufe erfolgt auf Ebene 2 (Application Service Implementations). Sie werden von Applikationsentwicklern erstellt. Die Interaktion mit Aktorik und Sensorik innerhalb der standardisierten Prozessabläufe erfolgt unter Nutzung der darunterliegenden Ebene 1 (Hardware Services). Ebene 1 wird von Bibliotheksentwicklern bereitgestellt und ermöglicht die Ansteuerung von Aktorik und Sensorik auf Grundlage einer auf dem System zur Verfügung gestellten Hardwareabstraktionsschicht.

Zur direkten Anbindung von Verwaltungsschalen werden spezialisierte Sprachelemente zur Verfügung gestellt, welche einen einfachen und gekapselten Zugriff auf Prozessdaten bieten. Dies ermöglicht, dass notwendige Prozessparameter (bspw. Regelparameter, Materialeigenschaften etc.) mittels der DSL gelesen und geschrieben werden können. Dadurch kann jede Abstraktionsebene der DSL mit Informationen aus externen Ressourcen parametriert bzw. konfiguriert werden. Diese sind auf der rechten Seite von Abbildung 2 dargestellt: Externe IT-Ressourcen, Steuerrezepte/Ablaufspezifikationen und Verwaltungsschalen der Produktionsmodule und Materialien. Die externen Ressourcen können beispielsweise Informationen zu Steuer- und Regelparametern, Stoffeigenschaften und elektrischen Konfigurationen bereitstellen. Die Parametrierung bzw. Konfiguration von Ebenen kann sowohl statisch als auch dynamisch erfolgen. Während eine statische Konfiguration zur Übersetzungszeit oder beim Deployment stattfindet, erfolgt die dynamische Variante zur Laufzeit des Produktionsprozesses.

Durch den Zugriff auf Prozessdaten in der Verwaltungsschale kann die Automatisierungssoftware flexibel auf Änderungen im Produktionsprozess reagieren. Dies ermöglicht weiterhin eine Variantenbildung der Automatisierungssoftware, um unterschiedliche Produktionsbedingungen und -parameter zu berücksichtigen. Beispielsweise kann die Software auf Änderungen von Steuer- und Regelparametern, Materialeigenschaften oder elektrischen Konfigurationen reagieren, indem sie entsprechend angepasst wird. Dies ermöglicht eine höhere Effizienz und Flexibilität im Produktionsprozess.

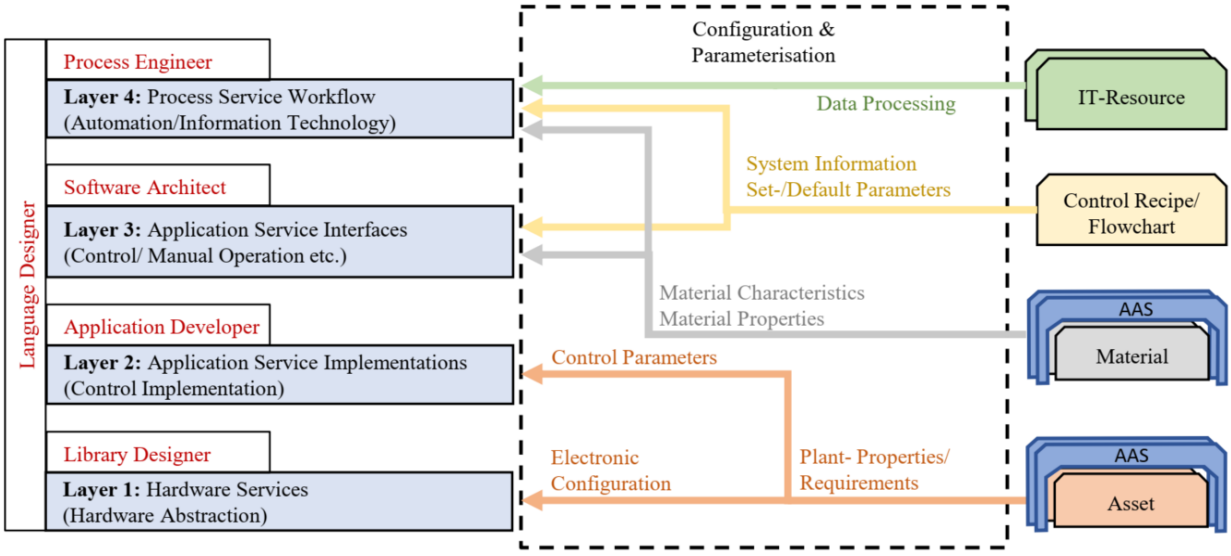

Abbildung 1: Konzept der hierarchischen DSL mit Abstraktionsebenen und der Anbindung an Verwaltungsschalen.

1.3.2.1 Implementierung der hierarchischen und serviceorientierten DSL

Die Implementierung des vorgestellten DSL-Konzepts erfolgte mit Hilfe des Software-Frameworks Eclipse Xtext. Da sowohl Eclipse Xtext als auch Eclipse Sirius das Eclipse Modeling Framework (EMF) einsetzen, ist eine Erweiterung der textuellen DSL zu einer grafischen Variante gewährleistet.

Nachfolgend werden beispielhaft Sprachkonstrukte der hierarchischen und serviceorientierten DSL tabellarisch aufgelistet. Hierbei werden zwischen Sprachelementen zur Definition von Ebenen (siehe Tabelle 1), zur Definition von Diensten/Funktionen (siehe Tabelle 2), zur Definition von Variablen und dem Aufruf von Diensten/Funktionen (siehe Tabelle 3) sowie zur Definition von Verzweigungen/Schleifenstrukturen (siehe Tabelle 4) unterschieden. Tabelle 5 zeigt die Sprachelemente zur Einbindung von Verwaltungsschalen, zur Variantenbildung und Definition von Nebenläufigkeiten.

Definition der Abstraktionsebenen	
Sprachelement	**Syntax**
Ebene 4: Process Service Workflow	**Layer_ProcessServiceWorkflow** *Name { … }*
Ebene 3: Application Service Interfaces	**Layer_AppServiceInterfaces** *Name { … }*
Ebene 2: Application Service Impl.	**Layer_AppServiceImplementations** *Name { … }*
Ebene 1: Hardware Services	**Layer_HardwareServices** *Name { … }*

Tabelle 1: Sprachelemente zur Definition von Abstraktionsebenen.

Definition von Diensten und Funktionen	
Betrachteter Layer	**Syntax**
Ebene 4: Process Service Workflow	**define processService DataType** Name (…)
Ebene 3: Application Service Inter.	**define interfaceService DataType** Name (…)
Ebene 2: Application Service Impl.	**define implementationService DataType** Name (…)
Ebene 1: Hardware Services	**define hardwareService DataType** Name (…)
Ebenen 1 - 4	**define function DataType** Name (…)

Tabelle 2: Sprachelemente zur Definition von Diensten und Funktionen.

Variablendeklaration/-definition und Dienste-/Funktionsaufrufe	
Sprachelement	**Syntax**
Variablendeklaration	**declare DataType** VariableName
Variablendefinition	**define DataType** VariableName = Ausdruck
Variablenzuweisung	**assign** VariableName = Ausdruck
Diensteaufruf	**call ServiceType** LayerName.serviceName (…)
Funktionsaufruf	**call function** functionName (…)
Aufruf von HAL-Diensten	**call halService** halServiceName (…)

Tabelle 3: Sprachelemente zur Variablendeklaration, Variablendefinition und zum Aufruf von Diensten/Funktionen.

Verzweigungen und Schleifen	
Sprachelement	**Syntax**
If-Verzweigung	**if (** Bedingung **) { … }**
While-Schleife	**while (** Bedingung **) { … }**
For-Schleife	**for (** Zählvariable, Ausdruck, Zählvariablenmanipulation**) { … }**

Tabelle 4: Sprachelemente zur Definition von Verzweigungen und Schleifenstrukturen.

Einbindung von Verwaltungsschalen, Servicegruppen und Definition von Nebenläufigkeiten	
Sprachelement	**Syntax**
Variablenaufruf aus AAS	**@inject** aasName **{ … }**
Variablensynchronisation aus AAS	**@databinding** variableName **{ … }**
Servicegruppendefinition	**define serviceGroup** Name**() { … }**
Servicegruppenaufruf	**call LayerName.**serviceGroupName(…)
Asynchroner Serviceaufruf	**call interfaceService async** ServiceName(…)
Asynchroner Serviceaufruf mit Priorität und Intervall	**call interfaceService async(**Priorität**) every(**xx**)** ServiceName(…)
Definition einer geteilten Variable	**define shared int** VariableName = Ausdruck
Kritischer Abschnitt	**criticalSection [ ]**

Tabelle 5: Sprachelemente zur Einbindung von Verwaltungsschalen, Servicegruppen und Definition von Nebenläufigkeiten.

1.3.2.2 Verarbeitungskette zur Zwischencode-Erzeugung und Interpreter-basierten Code-Ausführung

Die Hochschule Pforzheim erprobte die DSL anhand eines verfahrenstechnischen I4.0-Demonstrators. Nachfolgend wird die Verarbeitungskette zur Zwischencode-Erzeugung und -Ausführung mit dem Interpreter vorgestellt.

Das grundlegende Konzept des Ansatzes ist in Abbildung 2 dargestellt. Hierbei wird ausgehend von der Programmbeschreibung mittels der DSL ein abstrakter Syntaxbaum erzeugt. Diese Datenstruktur repräsentiert die hierarchische Aufteilung des textuellen Programmcodes und bietet dadurch die Grundlage zur Erzeugung eines Inter-Kontrollflussgraphen (Inter-Control Flow Graph). Dieser Kontrollflussgraph modelliert den konkreten Programmablauf und dient damit als Zwischencode für den Interpreter. Durch Überführen des Kontrollflussgraphen

in ein XML-Datenformat kann dieser Zwischencode dem DSL-Interpreter zur direkten Ausführung bereitgestellt werden.

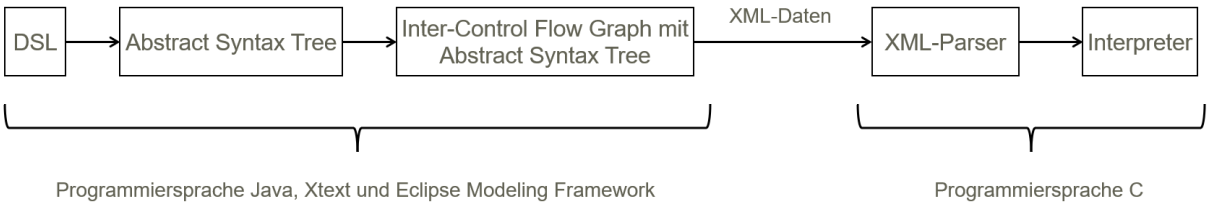

Abbildung 2: Verarbeitungskette zur Zwischencode-Erzeugung und Interpreter-basierten Code-Ausführung auf Grundlage der DSL.

1.3.2.3 Generierung von Inter-Control Flow Graphen als ausführbarer Zwischencode

Wie bereits im vorangegangenen Abschnitt erläutert, dient der auf Grundlage der DSL erzeugte Inter-Control Flow Graph (ICFG) als ausführbarer Zwischencode für den Interpreter. Er repräsentiert daher den konkreten Programmablauf. Der Graph besteht aus Knoten, welche einzelne Programmierausdrücke darstellen und welche mittels gerichteter Kanten miteinander verbunden sind. Folglich entsprechen Kanten dem Kontrollfluss des Programms. Da Programmierausdrücke komplexe arithmetische Ausdrücke beinhalten können, werden diese jeweils als abstrakte Syntaxbäume (AST) kodiert und den entsprechenden Knoten als Datenstruktur hinzugefügt. In Abbildung 3 wird die beschriebene Struktur eines Inter-Kontrollflussgraphen schematisch dargestellt.

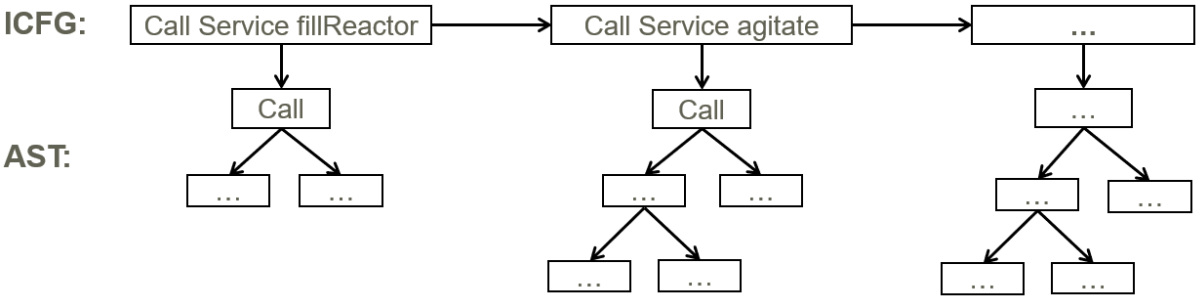

Abbildung 3: Exemplarische Darstellung eines Inter-Kontrollflussgraphen (ICFG) mit angehängten abstrakten Syntaxbäumen (AST).

Die softwaretechnische Modellierung des Inter-Kontrollflussgraphen erfolgt mittels eines EMF-Datenmodells. Grundlegend entspricht dieses Modell einer Adjazenzliste. Die einzelnen Programmierausdrücke der DSL werden mittels unterschiedlicher Knotenklassen modelliert. Darüber hinaus sind weitere Knoten zur Modellierung des Programmablaufes enthalten. Ein Überblick über das EMF-Datenmodell ist in Abbildung 4 dargestellt.

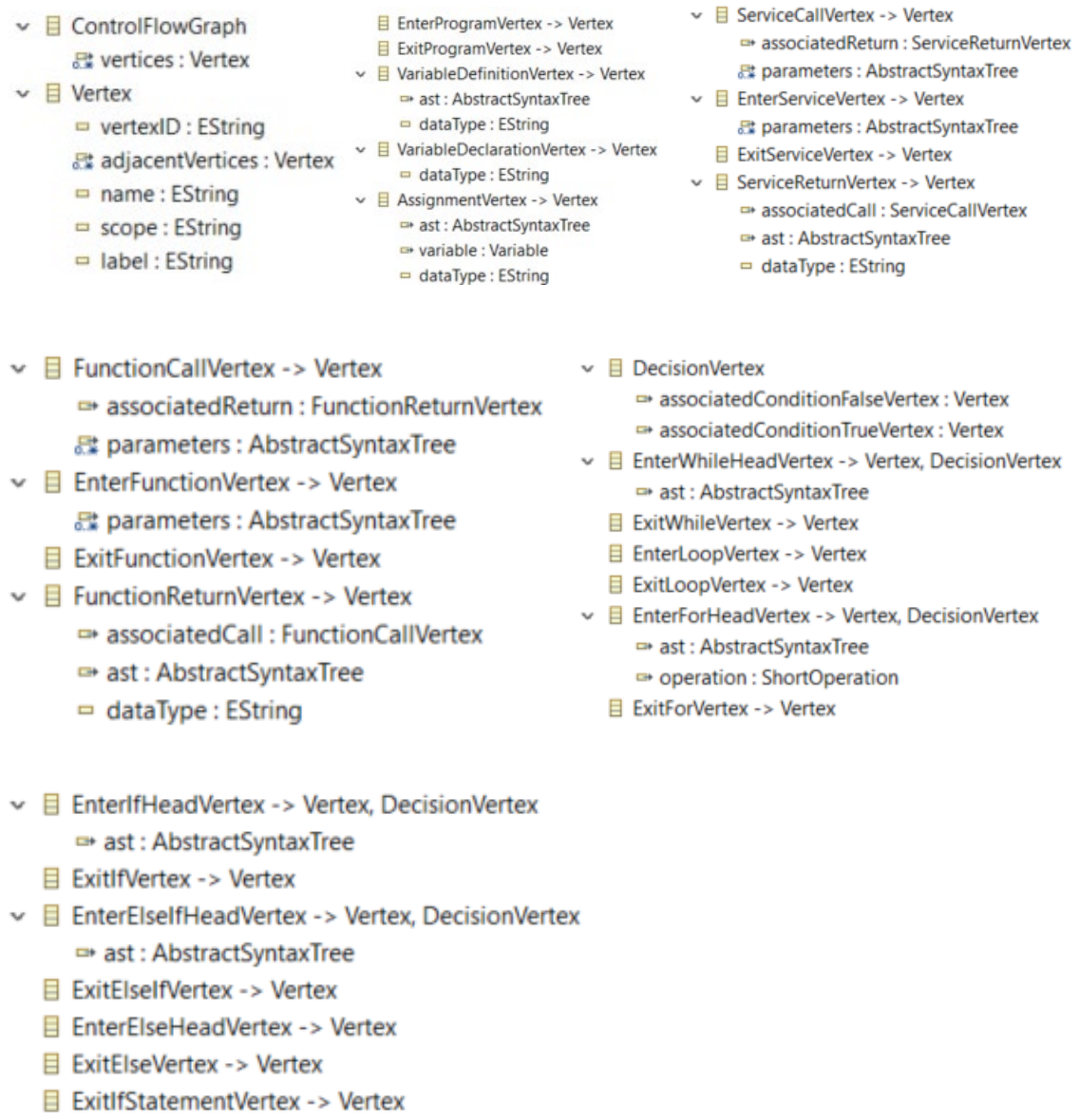

Abbildung 4: EMF-Datenmodell zur Modellierung von Inter-Kontrollflussgraphen.

Zur Erzeugung des erläuterten EMF-Datenmodells bzw. eines Inter-Kontrollflussgraphen wird die hierarchische Repräsentation des in der DSL verfassten Quellcodes in Form eines abstrakten Syntaxbaum genutzt. Diese Datenstruktur wird traversiert und die einzelnen Programmierausdrücke analysiert. Schlussendlich kann dadurch der Kontrollflussgraph erstellt werden.

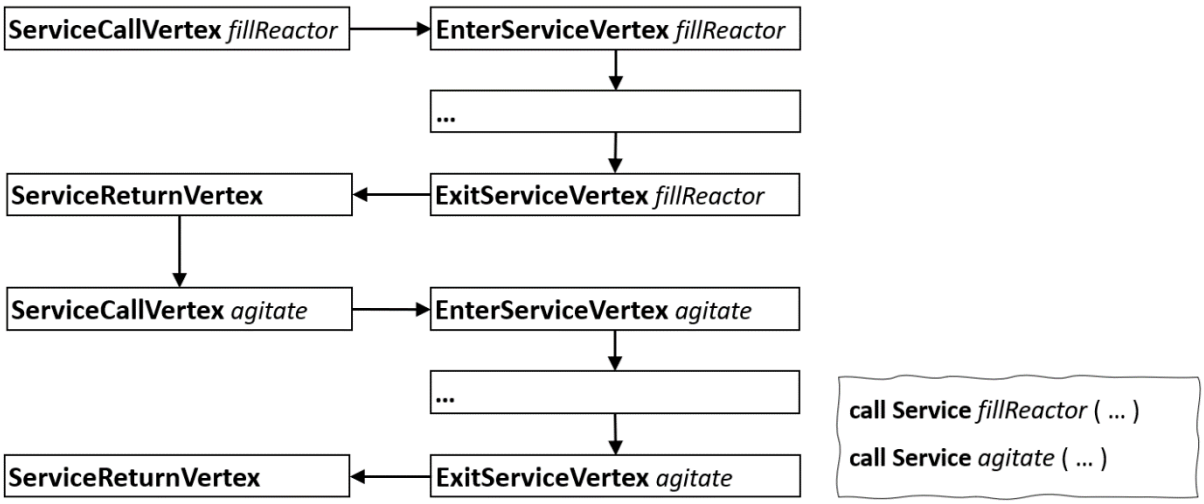

Abbildung 5: Inter-Kontrollflussgraph zum Aufruf von Prozessdiensten zur Umsetzung der verfahrenstechnischen Operationen *Befüllen* und *Vermischen*.

1.3.2.4 Ausführung nebenläufiger Programmteile

Bei dem Entwurf von nebenläufigen Programmen können eine Vielzahl von Fehlern entstehen, welche nur schwer nachvollziehbar sind. Neben typischen Fehlern in seriellen Programmabläufen wie zum Beispiel memory leaks, invalid page faults und buffer over reads, können zusätzlich Probleme bei Mehrfachzugriffen von geteilten Ressourcen auftreten, insbesondere Deadlocks, inkonsistente Zugriffe auf gemeinsame Variablen und Race Conditions.

Zur Modellierung von Automatisierungssoftware für verteilte Produktionssysteme ist die Bereitstellung nebenläufiger Softeare notwendig. Während moderne Programmiersprachen zur Anwendungsentwicklung, wie zum Beispiel Rust [1], spezielle Konzepte mit dem Ziel der Fehlervermeidung bei paralleler Programmausführung anbieten, sind entsprechende Mechanismen bei bestehenden Sprachen, bspw. gemäß IEC 61131-3, nicht vorhanden.

Im Rahmen der Entwicklung des Interpreters an der Hochschule Pforzheim wurde ein Konzept zur Ausführung nebenläufiger Programmteile erarbeitet. Der grundlegende Ansatz wird in Abbildung 6 veranschaulicht. Darin ist der Grundgedanke, dass die Nebenläufigkeit eines Programmes mittels paralleler Ausführungspfade innerhalb des Kontrollflussgraphen modelliert wird. Die einzelnen Ausführungspfade werden jeweils durch eine eigene Instanz des Interpreters abgearbeitet. Dies wird erreicht, indem die Interpreter-Instanzen auf jeweils einem eigenen Kernel-Thread des Betriebssystems auf dem System ausgeführt werden. Gemeinsame Daten, welche von mehreren Threads geschrieben und gelesen werden, sind in einen separaten Thread ausgelagert. Der Zugriff auf diese gemeinsamen Daten wird mit Hilfe von critical sections synchronisiert. Die Kommunikation zwischen Threads erfolgt mittels geteilter Variablen.

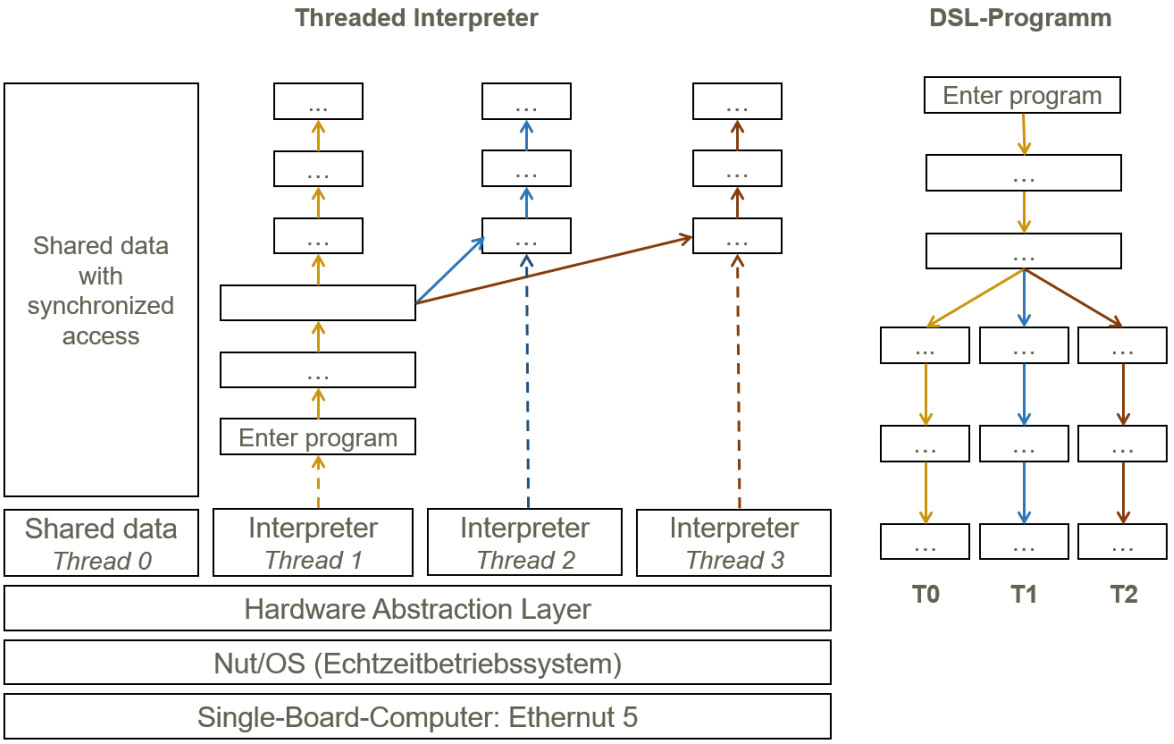

Abbildung 6: Konzept eines Interpreters zur Ausführung nebenläufiger DSL-Programme auf einer Hardwareplattform.

1.3.2.5 Umsetzung in ST-Code nach IEC 61131-3

Der Projektpartner Fa. logi.cals hat die entwickelte DSL und das Xtext-Framework genutzt, um ST-Code nach IEC 61131-3 zu generieren und in deren Produkt logi.CAD 3 zu integrieren Durch die Verwendung des EMF-Modells konnten zwischen den Projektpartnern gemeinsame Entwicklungsprototypen miteinander ausgetauscht werden.

1.3.2.6 Anwendbarkeit in der Fertigungstechnik

In Zusammenarbeit mit der Firma VSR Elektrotechnik wurde die Anwendbarkeit der grafischen Oberfläche in der Domäne der Fertigungstechnik überprüft. Dabei zeigte sich, dass die in Sirius programmierte grafische Oberfläche der DSL mit Ihrer Abstraktion der Fertigungsschritte sehr intuitiv und benutzerfreundlich ist und eine einfache Eingabe von Fertigungsparametern ermöglicht. Dies führt zu einer hohen Effizienz und Zuverlässigkeit im Produktionsprozess. Insbesondere die Möglichkeit, Fertigungsparameter visuell darzustellen, erlaubt es Anwendern, schnell und einfach Änderungen im Automatisierungsprogramm vorzunehmen.

Betrachteter Layer	Fertigungstechnisches Sprachelement
Ebene 4:	drillHole()
Ebene 3:	drillingControl()
Ebene 2:	setValueOfDrillingTool()

Ebene 2:	piDrillingController
Ebene 1:	setInputOfPWM
Ebene 1:	drillHole

Tabelle 6: Auswahl fertigungstechnischer Sprachelemente nach Ebenen aufgeschlüsselt.

Abbildung 7 zeigt, wie die fertigungstechnischen Sprachelemente aus Tabelle 7 grafisch programmiert werden.

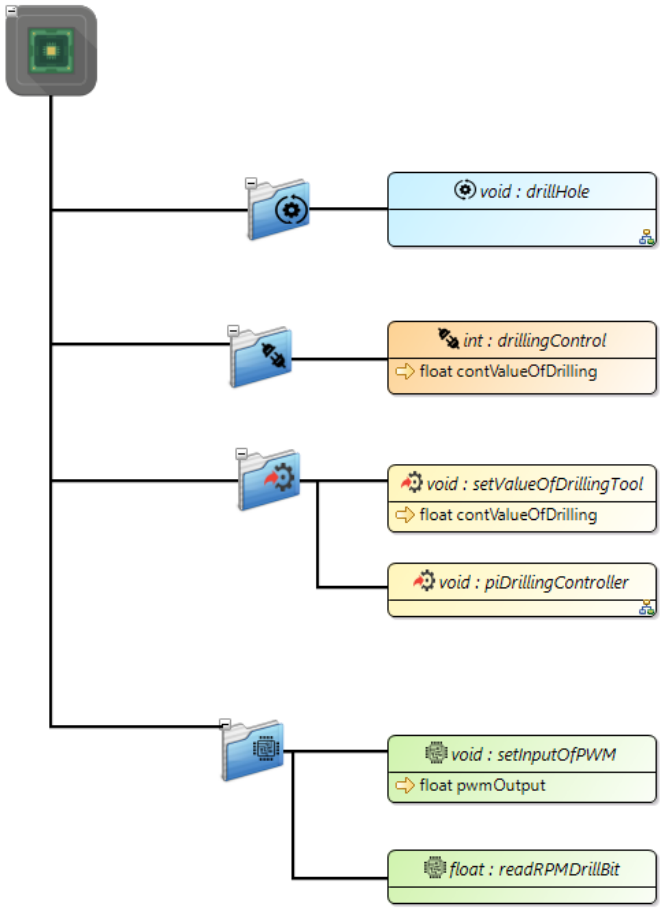

Abbildung 7: Darstellung der Services auf den vier Ebenen im grafischen Editor in der Domäne der Fertigungstechnik

1.3.3 Zeitverhalten des Interpreters

Am verfahrenstechnischen Versuchsdemonstrator 4.0-Minifab des Instituts, speziell dem Agitation-Modul wurden die Durchlaufzeiten der DSL hinsichtlich „Prozessechtzeit“ untersucht.

Zuvor wurden verschiedene Modultests durchgeführt, um bspw. die korrekte Generierung von Zwischencode zu überprüfen. Der Fokus der hier ausgeführten Messung, mithilfe eines Logic Analyzers, liegt auf den Ausführungszeiten der nebenläufigen Tasks im Interpreter. Das Anwendungsszenario beschreibt einen Ablauf, bei dem ein Behälter mit einem flüssigen Medium gefüllt und gerührt wird, wobei die Rührgeschwindigkeit mit steigendem Füllstand erhöht wird. Es werden zwei Interpreter-Tasks verwendet, die jeweils den aktuellen Füllstand des Behälters benötigen, der mittels eines Ultraschallsensors erfasst wird. Beide Interpreter-Tasks greifen auf den Sensor zu. Die Interpreter-Tasks werden periodisch ausgeführt und es wird überprüft, wie sich der Zugriff auf die geteilten Ressourcen auf die Ausführungszeiten auswirkt.

Bei der Messung wurden die Ausführungszeiten der beiden Interpreter-Tasks und des Idle-Tasks aufgezeichnet. Das Ergebnis zeigt, dass die periodische Ausführung der Interpreter-Tasks und die Erzeugung, Ausführung und Terminierung korrekt durchgeführt wurden. Anhand der während einer Messung aufgezeichneten Ausführungszeiten des Interpreter-Servicetasks kann diese Zeit näherungsweise bestimmt werden. Das arithmetische Mittel liegt bei 2,6023 ms und die Standardabweichung beträgt 0,00133 ms (gerundet auf fünf Nachkommastellen).

1.4 Andere wesentliche Ereignisse

Im Berichtszeitraum sind keine anderen wesentlichen Ereignisse aufgetreten.

2 Stand des Vorhabens

Die Bearbeitung des Vorhabens begann zum 01.11.2019 und wurde planmäßig am 30.09.2022 abgeschlossen.

		planmäßig begonnen	planabweichend begonnen	planmäßig abgeschlossen	planabweichend abgeschlossen		
Arbeitspaket	ID aus dem Arbeitsplan					Beginn	Ende
AP1: Anforderungsanalyse	AP1	☐	☒	☐	☒		
AP2: Domänenspezifische Sprache	AP2	☒	☐	☐	☒	1.1.20	31.12.21

AP3: Codegenerierung	AP3	☒	☐	☒	☐	1.4.20	30.6.22
AP4: Virtuelle Maschine	AP4	☒	☐	☒	☐	1.4.20	30.6.22
AP5: Evaluation	AP5	☒	☐	☒	☐	1.1.21	30.9.22
AP6: Projekt- und Qualitätsmanagement	AP6	☒	☐	☒	☐	1.10.19	30.9.22

2.1 Arbeits- und Zeitplanung

Zu Beginn kam es zu Verzögerungen bei der Einstellung eines geeigneten Mitarbeiters. Dies Verzögerung konnte im Verlauf des Vorhabens ausgeglichen werden.

2.2 Ausgabenplanung

Aufgrund der Covid-19-Pandemie war die Präsenz von Studierenden an der Hochschule in eingeschränkt. Daher war es nur eingeschränkt möglich, wissenschaftliche Hilfskräfte im Projekt einzusetzen. Entsprechend musste eine Reihe von Tätigkeiten durch Beschäftigte übernommen werden.

3 Erfolgsaussichten des Vorhabens

Innerhalb des Berichtszeitraums hat sich nichts ergeben, was der Erreichung der Projektziele entgegensteht.

4 Ergebnisse von dritter Seite

Es sind im Berichtszeitraum keine Ergebnisse von dritter Seite bekannt geworden, die für die Durchführung des Vorhabens relevant sind.

5 Änderung der Zielsetzung

Es ist keine Änderung der Zielsetzung erforderlich.

6 Fortschreibung des Verwertungsplans

Die Ziele des Vorhabens wurden vollständig erreicht. Zu Ende der Projektlaufzeit konnten bereits zwei thematisch verwandte, umfangreiche Nachfolgeprojekte akquiriert werden. Basierend auf den Projektergebnissen sind weitere Publikationen geplant, die sich insbesondere mit der Nutzung von Informationen aus der Verwaltungsschale zur Parametrisierung der

Automatisierungssoftware befassen. Auch das Thema Entwicklung von sicherer nebenläufiger Automatisierungssoftware soll weiter verfolgt werden.

6.1 Erfindungen/Schutzrechtsanmeldungen

Es wurden keine Erfindungen oder Schutzrechte im Berichtszeitraum angemeldet.

6.2 Wissenschaftliche und wirtschaftliche Erfolgsaussichten

Die wissenschaftlichen und wirtschaftlichen Erfolgsaussichten haben sich seit Projektbeginn nicht verändert. Die nachfolgend aufgeführten Ergebnisse bieten vielfältige Anwendungsmöglichkeiten:

- Das hierarchische Konzept der DSL erlaubt eine vereinfachte Zusammenarbeit von Experten für Automatisierungstechnik und Informatikern. Dies ermöglicht - gerade auch in Hinsicht auf den Fachkräftemangel in IT-Berufen - einen vereinfachten Ansatz zur Erstellung von Automatisierungssoftware.
- Die Integration von Steuerungsaufgaben in intelligente Sensoren und Aktoren erlaubt es, kleinere Steuerungsaufgaben zukünftig anders zu konzipieren. Er ergibt sich eine vereinfachte Automatisierungsarchitektur. Entsprechende Entwicklungen stehen erst am Anfang.
- Der Interpreter-basierte Ansatz erlaubt eine ressourcensparende Integration in intelligente Sensoren und Aktoren.
- Eine nebenläufige Ausführung der erstellten Software ist möglich.
- Das DSL-Konzept basiert auf dem Framework Eclipse Xtext und nutzt das Eclipse Modeling Framework (EMF). Damit eröffnen sich vielfältige Erweiterungsmöglichkeiten. Die Möglichkeit einer grafischen Programmierung und die Integration von ST-Code in Automatisierungssoftware für SPS-Steuerungen wurden im Rahmen des Vorhabens bereits aufgezeigt.
- Die Nutzung von Verwaltungsschalen zur Parametrisierung einer DSL stellt ein neues Konzept dar, dessen Möglichkeiten erst ansatzweise sichtbar werden. Hieraus ergibt sich ein hohes Potenzial für Folgeprojekte, siehe hierzu auch die Ausführungen über Folgeprojekte im Abschnitt 2.3.

Im Jahr 2021 wurde das Konzept der hierarchischen DSL unter Nutzung von Verwaltungsschalen auf der IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), die wichtigste internationale Konferenz für Fabrikautomatisierung, veröffentlicht [3].

Die Smart City Days in Pforzheim behandeln Themen der intelligenten Stadtentwicklung. Diese Entwicklungen sollen durch den Einsatz von Technologien wie dem Internet der Dinge (IoT), Künstlicher Intelligenz (KI) und Datenanalyse die Lebensqualität der Bürger verbessern. Im Rahmen dieser Veranstaltung im Oktober 2022 wurde die Nutzung von intelligenten Sensoren und Aktoren auf kleinen Automatisierungsplattformen durch eine vereinfachte Softwareentwicklung vorgestellt.

Ende 2022 wurden Erkenntnisse aus dem Projekt in einem Beitrag auf der renommierten Konferenz IEEE ICIEA (International Conference on Industrial Engineering and Applications) veröffentlicht. Der Beitrag befasst sich mit der automatischen Generierung von Automatisierungssoftware-Varianten in cyber-physischen Produktionssysteme basierend auf DSL und Informationen aus der Verwaltungsschale [4].

6.3 Wissenschaftliche und wirtschaftliche Anschlussfähigkeit

Digitalisierung der Produktion ist eines der strategischen Ziele des Instituts für Smart Systems und Services (IoS^3). Durch die im Rahmen des Projektes erarbeiten Lösungen und erworbenen Kompetenzen ergeben sich eine Reihe neuer Möglichkeiten für Folgeprojekte und Publikationen.

- Das ausgearbeitete Konzept der hierarchischen DSL ist herstellerunabhängig und kann einfach erweitert werden. Entsprechend bietet sich eine Anpassung der DSL für weitere Anwendungsdomänen an.
- Eine Erweiterung mit der Zielsetzung eine nachhaltige und energieeffiziente Produktion zu erreichen, ist im Rahmen des neu akquirierten Projektes „greenProd“ (Ausschreibung GreenTech des BMWK) geplant (s. u).
- Das hierarchische Konzept erlaubt den schnelleren und weniger fehleranfälligen Entwurf von Automatisierungssoftware.
- Das hierarchische Konzept erlaubt das vereinfachte Testen von Automatisierungssoftware.
- Die Möglichkeit Automatisierungssoftware ressourcenschonend in intelligente Sensoren und Aktoren zu integrieren, erlaubt den Entwurf einfacher Automatisierungslösungen, die ohne SPS aufgebaut werden können.
- Die Nutzung von Verwaltungsschalen zur Parametrisierung der DSL stellt ein neues Konzept dar. Im Beitrag auf der IEEE ICIEA wird eine Lösung zum flexiblen Umgang mit Softwarevarianten beschrieben. Eine solche Vorgehensweise erlaubt die schnellere Anpassung von Automatisierungssoftware an unterschiedlichen Anlagenkonfigurationen.
- Die Verbindung mit der Middleware BaSys (Fraunhofer IESE) erschließt einen großen Anwenderkreis.

Das Projekt hat bereits dazu beigetragen, zwei Folgeprojekte zu akquirieren. Zum einen handelt es sich um ein Projekt zum Entwurf intelligenter Sensoren und Aktoren in der Produktion. Dieses Projekt „KISA“ wird von der Carl-Zeiss-Stiftung gefördert (Projektvolumen ca. 700.000,- Euro). Zum anderen wird die Erweiterung der entwickelten DSL zur Automatisierung im Kontext einer nachhaltigen Produktion gefördert (Projekt „greenProd“, Ausschreibung GreenTech des BMWK, Projektvolumen ca. 1 Mio. Euro). Eingesetzt wird hierzu u. a. die Verwaltungsschale zur Parametrierung und zum Datenaustausch. Dieses neue Vorhaben soll dazu beitragen, eine vereinfachte nachhaltige Produktion zu ermöglichen und damit die Wettbewerbsfähigkeit der deutschen Industrie zu stärken. In dieser Ausschreibung werden nur 16,5 % der eingereichten Anträge gefördert.

7 Literatur

[1] W. Hemming und W. Wagner, Verfahrenstechnik, 11. Aufl., Vogel Business Media, 2011.

[2] S. Klabnik und C. Nichols: „The Rust Programming Language“, https://doc.rust-lang.org/book/#the-rust-programming-language, abgerufen am 28.04.21, 2018.

[3] C. Lehnert, G. Engel, H. Steininger, R. Drath and T. Greiner, "A Hierarchical Domain-Specific Language for Cyber-physical Production Systems Integrating Asset Administration Shells," 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, 2021, pp. 01-04, doi: 10.1109/ETFA45728.2021.9613428.

[4] C. Lehnert, G. Engel and T. Greiner, "A Hierarchical Domain-Specific Language Supporting Variants of CPPS Software and Integrating Asset Administration Shells," 2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA), Chengdu, China, 2022, pp. 572-577, doi: 10.1109/ICIEA54703.2022.10006111.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

Kurzbericht
IMPACT

Innovative Methods for Programming of Automation Control Technology

Hochschule Pforzheim, Institut für Smart Systems und Services, Tiefenbronner Str. 65, 75175 Pforzheim
Berichtszeitraum: 1.10.2019 – 30.09.2022
Förderkennzeichen: 01IS19031B
Laufzeit: 1.10.2019 – 30.09.2022

Das Projekt "IMPACT" hat sich zum Ziel gesetzt, Software zur Automatisierung der Produktion zu vereinfachen und zu optimieren. Hierfür wurde eine hierarchische und serviceorientierte domänenspezifische Sprache (DSL) entwickelt, die es ermöglicht, Automatisierungssoftware intuitiv und effizient zu entwickeln. Neben Logik- und Datenstrukturen bietet diese DSL je nach Anwendungsdomäne (Prozesstechnik oder Fertigungstechnik), passende Sprachkonstrukte. Weiterhin werden spezialisierte Modellierungselemente zur direkten Einbindung von Verwaltungsschalen und zur Ausführung von verteilten Automatisierungsdiensten bereitgestellt. Hierzu werden spezifische Diensteschnittstellen und Kommunikationsprotokolle genutzt und vorhandene System- und Prozessobjekte berücksichtigt.

Das hierarchische Konzept der DSL beruht auf vier aufeinander aufbauenden Ebenen, die eine granulare Verfeinerung bzw. Konkretisierung des Programmablaufs zulassen. Die Ebenen sind strikt hierarchisch organisiert, d.h. es können keine Ebenen übersprungen werden. Weiterhin unterscheiden sich die Ebenen in der bereitgestellten Funktionalität und der Abstraktionsgrad der Ebenen. Dieser nimmt zu tieferen Ebenen hin ab. Ebene 4 (Process Service Workflow) bietet den höchsten Abstraktionsgrad und dient zur Definition des grundlegenden Prozessablaufs. Dieser wird von einem Prozessingenieur mittels einzelnen aufeinanderfolgenden Prozessoperationen bzw. Fertigungsschritten, abgeleitet von einem Rezept oder einem Ablaufplan, erstellt. Dazu nutzt er von Softwarearchitekten entwickelte Funktionsbausteine, welche auf Ebene 3 (Application Service Interfaces) bereitgestellt werden. Diese Funktionsbausteine bilden die Schnittstelle zu Steuerungen/Regelungen für standardisierte Prozessabläufe. Die konkrete Implementierung dieser Abläufe erfolgt auf Ebene 2 (Application Service Implementations). Sie werden von Applikationsentwicklern erstellt. Die Interaktion mit Aktorik und Sensorik innerhalb der standardisierten Prozessabläufe erfolgt unter Nutzung der darunterliegenden Ebene 1 (Hardware Services). Ebene 1 wird von Bibliotheksentwicklern bereitgestellt und ermöglicht die Ansteuerung von Aktorik und Sensorik auf Grundlage einer auf dem System zur Verfügung gestellten Hardwareabstraktionsschicht.

Durch den Zugriff auf Prozessdaten in der Verwaltungsschale kann die Automatisierungssoftware flexibel auf Änderungen im Produktionsprozess reagieren. Dies ermöglicht weiterhin eine Variantenbildung der Automatisierungssoftware, um unterschiedliche Produktionsbedingungen und -parameter zu berücksichtigen. Beispielsweise kann die Software auf Änderungen von Steuer- und Regelparametern, Materialeigenschaften oder elektrischen Konfigurationen

reagieren, indem sie entsprechend angepasst wird. Diese Vorgehensweise erlaubt eine hohe Effizienz und Flexibilität im Produktionsprozess.

Im Rahmen der Entwicklung des Interpreters an der Hochschule Pforzheim wurde ein Konzept zur Ausführung nebenläufiger Programmteile erarbeitet. Der Grundgedanke liegt darin, dass die Nebenläufigkeit eines Programmes mittels paralleler Ausführungspfade innerhalb des Kontrollflussgraphen modelliert wird. Die einzelnen Ausführungspfade werden jeweils durch eine eigene Instanz des Interpreters abgearbeitet. Dies wird erreicht, indem die Interpreter-Instanzen auf jeweils einem eigenen Kernel-Thread des Betriebssystems auf dem System ausgeführt werden. Gemeinsame Daten, welche von mehreren Threads geschrieben und gelesen werden, werden in einen separaten Thread ausgelagert. Der Zugriff auf diese gemeinsamen Daten wird mit Hilfe von critical sections synchronisiert. Die Kommunikation zwischen Threads erfolgt mittels geteilter Variablen.

Die Implementierung des vorgestellten DSL-Konzepts erfolgte mit Hilfe des Software-Frameworks Eclipse Xtext. Da sowohl Eclipse Xtext als auch Eclipse Sirius das Eclipse Modeling Framework (EMF) einsetzen, ist eine Erweiterung der textuellen DSL zu einer grafischen Variante gewährleistet.

Abschließend wurde die Evaluation der DSL und des Interpreters an der I4.0 Modellanlage des Instituts für Smart Systems und Services an der HS Pforzheim erfolgreich durchgeführt.