22. AALE-Konferenz
Rosenheim, 25.–27. Februar 2026

Die AAS als Schlüssel zur ganzheitlichen Steuerungscodegenerierung: Ein Diskussionspapier

Andreas Würger[1]

Zusammenfassung

Die manuelle Erstellung von Steuerungscode ist zeitaufwendig und fehleranfällig. Automatische Codegenerierung kann diesen Aufwand reduzieren, benötigt dafür jedoch viele Informationen, die bisher meist manuell bereitgestellt oder in komplexen Modellen gepflegt werden müssen. In diesem Beitrag wird untersucht, wie die Asset Administration Shell (AAS) als digitales Anlagenmodell diese Nachteile verringern kann. Es wird gezeigt, dass AAS-basierte Modelle über den Anlagenlebenszyklus hinweg automatisch entstehen und Engineering-Werkzeugen eine einheitliche und aktuelle Informationsbasis für die Codegenerierung bieten. Dadurch lassen sich viele wiederkehrende Aufgaben einfacher automatisieren. Gleichzeitig bleibt vollständig automatisch erzeugter Steuerungscode weiterhin schwierig, da dafür zusätzliche Kontextinformationen fehlen. Abschließend wird diskutiert, inwieweit zukünftige LLM-basierte Ansätze helfen können, diese verbleibende Lücke zu schließen.

Stichwörter

Automatische Steuerungscodegenerierung, digitaler Zwilling, Asset Administration Shell

1 Einleitung

Die Erstellung von Steuerungscode nimmt einen hohen Anteil am Engineering-Aufwand automatisierungstechnischer Anlagen ein. Gleichzeitig ist die manuelle Steuerungscodeerstellung fehleranfällig. Eine Möglichkeit Aufwand und Fehlerquellen zu reduzieren ist die automatische Steuerungscodegenerierung.

In diesem Beitrag wird diskutiert wie die AAS-Technologie, bzw. AAS-basierte Anlagenmodelle als Informationsbasis für automatische Codegenerierung genutzt werden können und welche Vorteile sich gegenüber bisherigen Modellierungstechnologien wie z. B. AutomationML [1] ergeben. Des Weiteren soll aufgezeigt werden, welche völlig neuen Möglichkeiten für die automatische Generierung von Steuerungscode sich aus den Eigenschaften der AAS-Technologie heraus ergeben.

Hierzu wird zunächst der aktuelle Stand von Technik und Forschung hinsichtlich automatischer Codegenerierung analysiert. Anschließend wird eine kurze Einführung in die AAS-Technologie gegeben. In dem darauffolgenden Abschnitt werden zwei Hypothesen für diese Arbeit aufgestellt, welche im Folgenden belegt werden. Den Abschluss des Beitrags bildet ein Fazit darüber, inwieweit die in dieser Arbeit vorgestellten Ansätze die bisherigen Nachteile etablierter Steuerungscodegenerie-

[1] Prof. Dr.-Ing. Andreas Würger, Professor für industrielle Steuerungs- und Regelungstechnik, E-Mail: andreas.wuerger@haw-hamburg.de, HAW-Hamburg, Fakultät Elektro-, Medien- und Informationstechnik, Berliner Tor 7, 20099 Hamburg

rungsverfahren mitigieren und inwieweit nicht und wie verbleibende Nachteile in der Zukunft endgültig mitigiert werden könnten.

2 Stand von Technik und Forschung: SPS-Codegenerierung

Ein generisches SPS-Programm kann aus den folgenden Programmteilen bestehen:

- Ablauflogik: Steuerung und Koordination des Prozessablaufs durch Zustandsautomaten, Schrittketten oder Ablaufsteuerungen, die den sequentiellen Ablauf von Maschinen- oder Anlagenfunktionen definieren.
- Schnittstellen- oder domänenspezifische Logik: Instanzen von Funktionsbausteinen aus schnittstellen- oder domänenspezifischen Funktionsbausteinbibliotheken, die die Kommunikation mit anderen Systemen oder die Umsetzung branchenspezifischer Anforderungen ermöglichen.
- Fehlererkennung und Alarmgenerierung: Überwachung von Prozess- und Systemzuständen zur Identifikation von Störungen oder Abweichungen sowie Generierung von Meldungen, Warnungen oder Alarmen zur Unterstützung der Fehlerdiagnose und -behebung.
- Verriegelungslogik: Sicherstellung sicherer Betriebszustände durch logische Verknüpfungen, die kritische Aktionen nur bei erfüllten Bedingungen (keine Alarme anstehend, bestimmte Variablen gesetzt/ nicht gesetzt) zulassen.
- Aufrufe und Verknüpfungen:
  - Mapping von Ein- und Ausgangsvariablen auf Rangiervariablen.
  - Strukturiertes Aufrufen von Funktionsbausteinen und Programmteilen.
  - Verknüpfung der Programmteile durch entsprechende Variablen.

Die automatische Generierung von Steuerungscode kann grob in zwei Verfahren unterteilt werden:

1. Die Generierung eines Ablauflogikcodes durch Transformation aus einem geeigneten Modell (z. B. aus einem steuerungstechnisch interpretierbaren Petri Netz oder aus einem GRAFCET-Modell heraus).
2. Die Generierung von aufgabenorientiertem Code durch Zusammensetzung des Codes aus vorgefertigten Code-Snippets (z. B. die Generierung von SPS-Programmteilen für Energiemanagementaufgaben).

Nach aktuellem Stand von Forschung und Technik existieren für jeden der zuvor genannten Programmteile Methoden, um diese nach einem der beiden genannten Verfahren der automatischen Steuerungscodegenerierung zu generieren.

[2, 3 und 4] stellen Konzepte zur automatischen Generierung von Steuerungscode aus steuerungstechnisch interpretierbaren Petrinetzen [5] (SIPN) vor. Ein Algorithmus zur Prozesssteuerung wird dabei durch ein SIPN beschrieben. Dieses wird anschließend in eine IEC-61131-3-Sprache übersetzt.

[6] stellt ein Verfahren vor, das in der Spezifikationssprache GRAFCET [7] beschriebene Abläufe in Steuerungscode umwandelt. In [8] wird ein Konzept zur bidirektionalen Transformation zwischen GRAFCET-Spezifikationen und Steuerungscode vorgestellt.

[9] beschriebt wie Programmteile für Energiemanagementaufgaben durch Zusammensetzung des Codes aus vorgefertigten Code-Snippets und Instanziierung von vorgefertigten Funktionsbausteinen generiert werden.

[10] beschreibt ein Verfahren zur automatischen Generierung von Verriegelungslogik. Dabei wird auf Komponenten verfahrenstechnischer Anlagen wie z. B. Pumpen, Behälter, Ventile, etc. fokussiert.

Es existieren auch Ansätze für eine ganzheitliche Steuerungscodegenerierung, bei denen ein Codegenerator dann ein komplettes (fertiges, lauffähiges) Programm generiert. So wird z. B. in [11] ein

Ansatz zur möglichst vollständigen Generierung von Steuerungscode vorgestellt. Hierbei werden möglichst viele wiederkehrende Codefragmente in Funktionsbausteinen gekapselt und in Bibliotheken abgelegt. In [12] wird ein auf diesem Ansatz aufbauendes wissensbasiertes Verfahren zur Generierung von Steuerungscode für Fertigungsanlagen vorgestellt.

Ein weiterer aktueller Forschungsansatz ist der Einsatz großer Sprachmodelle (LLMs) für die automatische Generierung von SPS-Code. Derzeit können LLMs jedoch noch keinen zuverlässig einsetzbaren PLC-Code (Structured Text, ST) erzeugen. Gründe dafür sind das Fehlen offener Trainingsdatensätze, die hohe Komplexität industrieller Steuerungslogik sowie typische Probleme generischer LLMs wie Halluzinationen und Funktionsfehler. Neue Ansätze wie in [13] versuchen diese Grenzen durch gezielte Prompt-Strukturen und zusätzliche Logikmodule zu verringern und zeigen erste Fortschritte in Richtung KI-gestützter Codegenerierung.

Die Ansätze für eine ganzheitliche Steuerungscodegenerierung benötigen als Eingangsgröße große Mengen an Informationen. Diese Informationen müssen entweder manuell (z. B. über eine Benutzerschnittstelle) in den Codegenerator eingegeben werden oder sind in entsprechend komplexen Informationsmodellen (z. B. in AutomationML) abgelegt, welche dann in den Codegenerator eingelesen werden. Durch die entsprechend aufwendige Informationseingabe (in den Codegenerator selbst oder in das Modell) wird der der automatischen Generierung entgegenstehende Programmieraufwand für die konventionelle Programmerstellung nur teilweise mitigiert. Für eine vollständige Mitigation müssten also auch die für die Codegenerierung benötigten Informationen automatisch bezogen werden, bzw. das zugrunde liegende Modell automatisch entstehen.

3 Stand der Technik: Asset Administration Shell

Die AAS-Technologie (Asset Administration Shell, Verwaltungsschale) bildet den digitalen Zwilling eines Assets ab. Ein Asset kann hierbei eine Einzelkomponente, ein Feldgerät, eine Anlage, Software oder jedes andere Objekt sein, welches für eine Organisation einen Wert darstellt. Die Technologie wird von der Industrial Digital Twin Association e. V. (IDTA) als firmenübergreifendes Konsortium betreut und weiterentwickelt; das allgemeine AAS-Metamodell ist zudem als IEC-Norm [14] standardisiert.

Ein grundlegendes Prinzip der AAS ist die Unterscheidung zwischen Typ-AAS und Instanz-AAS, analog zu Klassen-Objekt-Beziehungen in der Softwareentwicklung. Typ-AASs beschreiben Asset-Typen, während Instanz-AASs konkrete existierende Einzel-Assets repräsentieren und üblicherweise während deren Produktion erzeugt werden. Häufig wird zusammen mit der Instanz-AAS eines Assets ein spezieller QR-Code [15] erzeugt, über dem auf die Instanz-AAS zugegriffen werden kann. AASs können verteilt auf mehreren AAS-Servern liegen; Verknüpfungen zwischen AASs sowie zwischen AASs und ihren Submodellen erfolgen über definierte Referenzen.

Die Informationen in den AASs enthaltenen Informationen werden strukturiert in standardisierten Teilmodellen (Submodels) abgelegt. Diese anwendungsorientierten Teilmodelle bilden sowohl grundlegende als auch spezifische Use Cases ab. Die IDTA führt dafür aktuell (Stand Dezember 2025) rund 100 veröffentlichte bzw. in Entwicklung befindliche Teilmodelle. Zu den verbreiteten Anwendungsfällen gehören:

- Das digitale Typenschild (digital Nameplate).
- Papierlose Dokumentation durch Ablegen von digitalen Dokumentationsunterlagen in der AAS.
- Bereitstellen von technischen Produktmerkmalen in der AAS.
- Bereitstellen des CO2-Fußabdrucks in der AAS.

Darüber hinaus unterstützt die AAS-Technologie zentrale Engineering-Use-Cases [16], wie z. B.:

- Management von (Automatisierungs-)Software-Modulen mit AASs.
- Beziehen und Wiederverwenden von Engineering-Daten die sich in der Anlagen-AAS befinden.
- Verwenden von Engineering-relevanten Daten aus den Komponenten-AASs
- Speicherung der beim SPS-Engineering erzeugten Engineering-Daten in AASs zur Weiterverwendung in anderen Engineering-Werkzeugen.

Für die Darstellung von zu Assets gehörender Software existiert das Teilmodell „Software Nameplate“ [17]. Das Teilmodell dient der standardisierten und interoperablen Bereitstellung von Softwareinformationen. Es besteht aus einem typbezogenen Aspekt (SoftwareNameplateType) mit Angaben wie Produktbezeichnung, Version, Hersteller oder Installationsquelle sowie einem instanzbezogenen Aspekt (SoftwareNameplateInstance) mit Details wie Seriennummer, Installationspfad, etc. Softwarepakete können im Typ-Aspekt direkt abgelegt oder über externe Bezugsquellen referenziert werden. Dieses Teilmodell bildet die Grundlage für die eindeutige Beschreibung, Versionierung und Verwaltung von Software innerhalb von AAS-Strukturen.

Komplexe Anlagenstrukturen können über die AAS durch das standardisierte Teilmodel „Hierarchical Structures“ abgebildet werden, welches hierarchische Beziehungen durch Referenzen auf unterlagerte AASs ermöglicht. Damit schafft die Technologie die Grundlage für ein über den gesamten Lebenszyklus automatisch erzeugbares und durchgängiges Anlagenmodell. Die AAS ist dabei bewusst domänen- und herstellerunabhängig gestaltet, um Interoperabilität über Branchen hinweg zu gewährleisten. Bild 1 zeigt ein, aus hierarchisch angeordneten AASs bestehendes, Anlagenmodell.

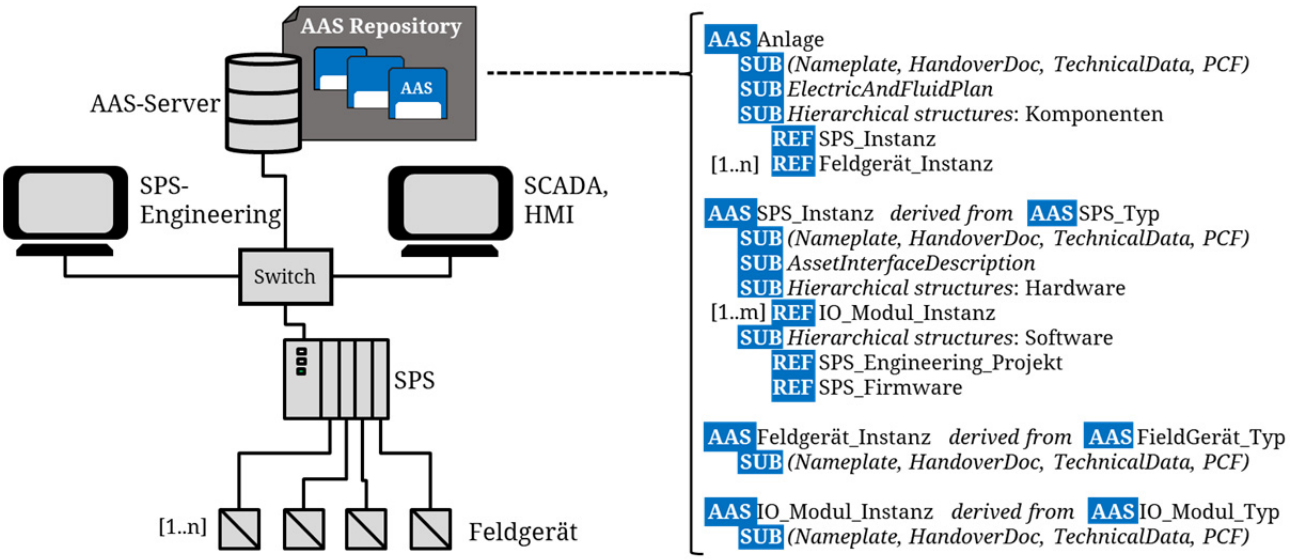

Bild 1: AAS-Hierarchie einer Anlage

AASs werden in AAS-Repositorys auf AAS-Servern gehostet. Der Zugriff erfolgt über eine REST-API, wobei Sicherheitsmechanismen wie feingranulares Rechtemanagement sowie Authentifizierung und Verschlüsselung bereits integriert sind.

4 Arbeitshypothesen

Aktuelle Ansätze zur ganzheitlichen Generierung von Steuerungscode benötigen als Eingangsgröße umfangreiche Informationen. Diese müssen heute entweder manuell in den Codegenerator eingegeben werden oder liegen in komplexen Informationsmodellen wie AutomationML vor, die zunächst erstellt und anschließend in den Codegenerator eingelesen werden müssen. Der damit verbundene Aufwand reduziert den Programmieraufwand der konventionellen Codeerstellung nur teilweise. Eine vollständige Mitigation wäre nur erreichbar, wenn die für die Codegenerierung benötigten Informationen automatisch bereitgestellt werden oder das zugrunde liegende Modell automatisch entsteht.

Auf Basis dieser Ausgangssituation werden die folgenden beiden Arbeitshypothesen formuliert:

Hypothese 1

Mit der AAS-Technologie kann ein Anlagenmodell dezentral und automatisch über den gesamten Anlagenlebenszyklus hinweg entstehen.

Da Instanz-AASs bereits während der Produktion erzeugt werden und im Betrieb weiter aktualisiert werden können, entsteht das Anlagenmodell nicht als einmalig zu erstellendes zentrales Modell, sondern als kontinuierlich wachsendes, stets aktuelles Abbild der realen Anlage. Die Informationen verteilen sich dabei auf die AASs der einzelnen Komponenten und werden über standardisierte Submodelle strukturiert bereitgestellt.

Hypothese 2

Engineering-Werkzeuge können die für die Codegenerierung notwendigen Informationen direkt aus dem dezentralen, lebenszyklusaktuellen Anlagenmodell beziehen.

Da alle relevanten Daten in standardisierten Submodellen hinterlegt und auf AAS-Servern abgelegt sind, können unterschiedliche Engineering-Werkzeuge direkt auf dieselben Informationen zugreifen. Diese können Daten sowohl beziehen als auch aktualisieren, sodass das Anlagenmodell konsistent bleibt und sich über den gesamten Anlagenlebenszyklus hinweg weiterentwickelt. Im Vergleich zu rein dateibasierten Ansätzen wie AutomationML bietet die AAS somit den entscheidenden Vorteil eines serverbasierten, kollaborativ nutzbaren Informationsmodells. AutomationML bleibt dennoch ein wichtiges Format für die strukturierte Ablage von Engineering-Daten, auch innerhalb der AAS.

5 Anlagenmodell über den Anlagenlebenszyklus

Das AAS-basierte Anlagenmodell entsteht nicht als zentrales, einmalig zu erstellendes Dokument, sondern entwickelt sich kontinuierlich über den gesamten Anlagenlebenszyklus hinweg. Bild 2 zeigt schematisch, wie die Anlagen-AAS bereits in der frühen Planungsphase angelegt wird und anschließend mit Informationen aus allen Engineering- und Betriebsphasen angereichert wird.

Zu Beginn der Anlagenplanung werden nacheinander Typ-AAS und Instanz-AAS erzeugt. Ab diesem Zeitpunkt werden sämtliche im Engineering entstehenden Daten in der Anlagen-AAS abgelegt. Dazu gehören z. B. Schaltpläne, Rohrpläne, Bestückungspläne, Konstruktionszeichnungen, Konfigurationsdateien oder andere Dokumentationsunterlagen. Auch softwarebezogene Informationen, wie etwa Versionsdaten oder Softwarepakete, die im Teilmodell „Software Nameplate“ strukturiert abgelegt werden können, tragen zu dieser Informationsbasis bei.

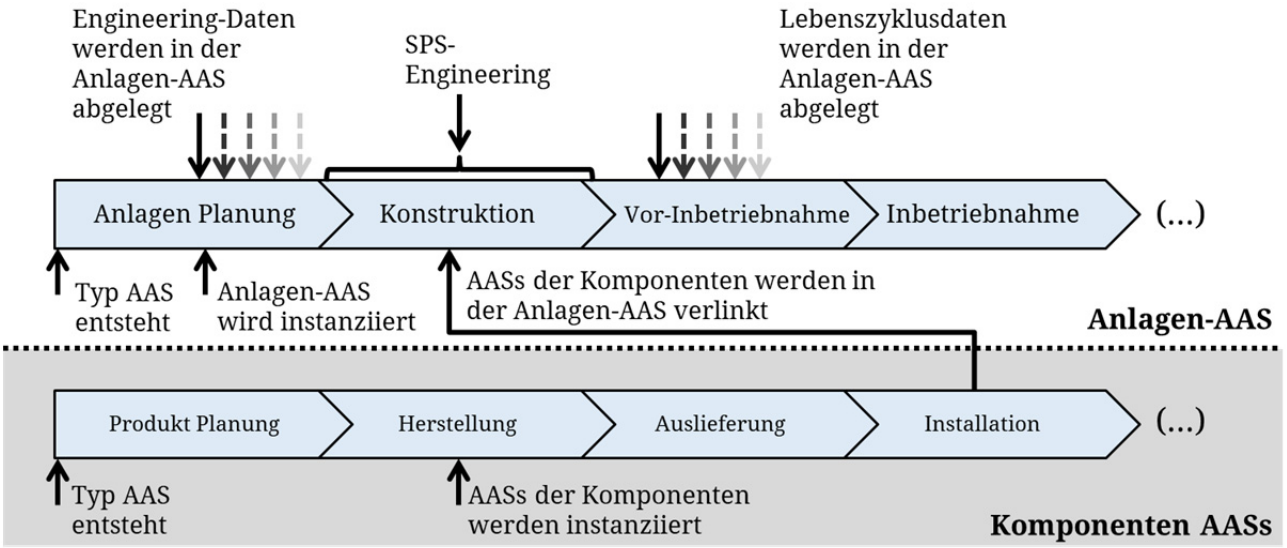

Bild 2: AAS-basiertes Anlagenmodell über den Anlagenlebenszyklus, angelehnt an [16]

Während der Konstruktion und des Detailengineerings werden die AASs der einzelnen Hardware- und Softwarekomponenten instanziiert und über das Teilmodell „Hierarchical Structures“ in der Anlagen-AAS referenziert. Auf diese Weise entsteht über die Zeit eine vollständige, hierarchische Abbildung der Anlage, die sowohl die reale Struktur der Komponenten als auch deren Engineering-Historie widerspiegelt. Dabei können beispielsweise komplexere Feldgeräte bereits in ihrer AAS die Informationen bereitstellen, die für ihre Integration in das SPS-Engineering notwendig sind. Feldgeräte, die über ein Kommunikationssystem angeschlossen sind, stellen in ihrer AAS zudem die vollständige Konfiguration ihrer Kommunikationsschnittstelle bereit. Solche Daten müssen im späteren SPS-Engineering nicht mehr manuell eingegeben oder rekonstruiert werden, sondern fließen automatisch in das entstehende Anlagenmodell ein.

Spätestens vor der Inbetriebnahme existiert ein geschlossenes Informationsmodell der gesamten Anlage. Mit zunehmender Betriebsdauer wird die Anlagen-AAS weiter durch Lebenszyklusdaten ergänzt, etwa durch Prüfprotokolle, Wartungsinformationen oder revisionsrelevante Dokumente. Damit bildet die Anlagen-AAS ein konsistentes, aktuelles und umfassendes Abbild der realen Anlage über alle Lebensphasen der Anlage hinweg.

Durch diese kontinuierliche Anreicherung der Anlagen-AAS entsteht automatisch genau die Informationsbasis, die für die in Abschnitt 2 beschriebenen Verfahren der automatischen Steuerungscodegenerierung benötigt wird. Engineering-Daten, die traditionell separat abgelegt werden und manuell in Codegeneratoren eingegeben werden müssten, entstehen im natürlichen Engineering-Prozess und werden direkt in der AAS abgelegt. Dadurch stehen beispielsweise Strukturdaten für die Generierung von Ablauflogikmodellen oder gerätespezifische Informationen zur Konfiguration von Funktionsbausteinen unmittelbar zur Verfügung. Die in Hypothese 1 formulierte Annahme, dass die AAS-Technologie ein über den Lebenszyklus automatisch wachsendes Anlagenmodell bereitstellt, wird hierdurch bestätigt.

Auch während des SPS-Engineerings erzeugte Daten – wie beispielsweise die Konfiguration eines OPC UA Servers – können direkt in der AAS abgelegt werden, um später zur Konfiguration überlagerter Systeme (HMI, SCADA, MES) wiederverwendet zu werden. Dadurch wird das Anlagenmodell zusätzlich erweitert und bleibt über alle Gewerke hinweg konsistent.

6 Werkzeugübergreifende Datennutzung

Die AAS-Technologie ermöglicht nicht nur die Entstehung eines über den Lebenszyklus hinweg automatisch wachsenden Anlagenmodells, sondern auch den serverbasierten Zugriff unterschiedlicher Engineering-Werkzeuge auf dieses Modell. Damit unterscheidet sich die AAS von rein dateibasierten Ansätzen, wie sie beispielsweise mit AutomationML verwendet werden. Während dateibasierte Modelle immer offline, verteilt, versionsabhängig und manuell gepflegt werden müssen, werden AASs in AAS-Repositorys auf AAS-Servern betrieben und können von mehreren Engineering-Werkzeugen verschiedener Gewerke parallel genutzt werden. Bild 3 stellt den gleichzeitigen Zugriff verschiedener Engineering-Werkzeuge/ Gewerke auf ein AAS-Repository dar.

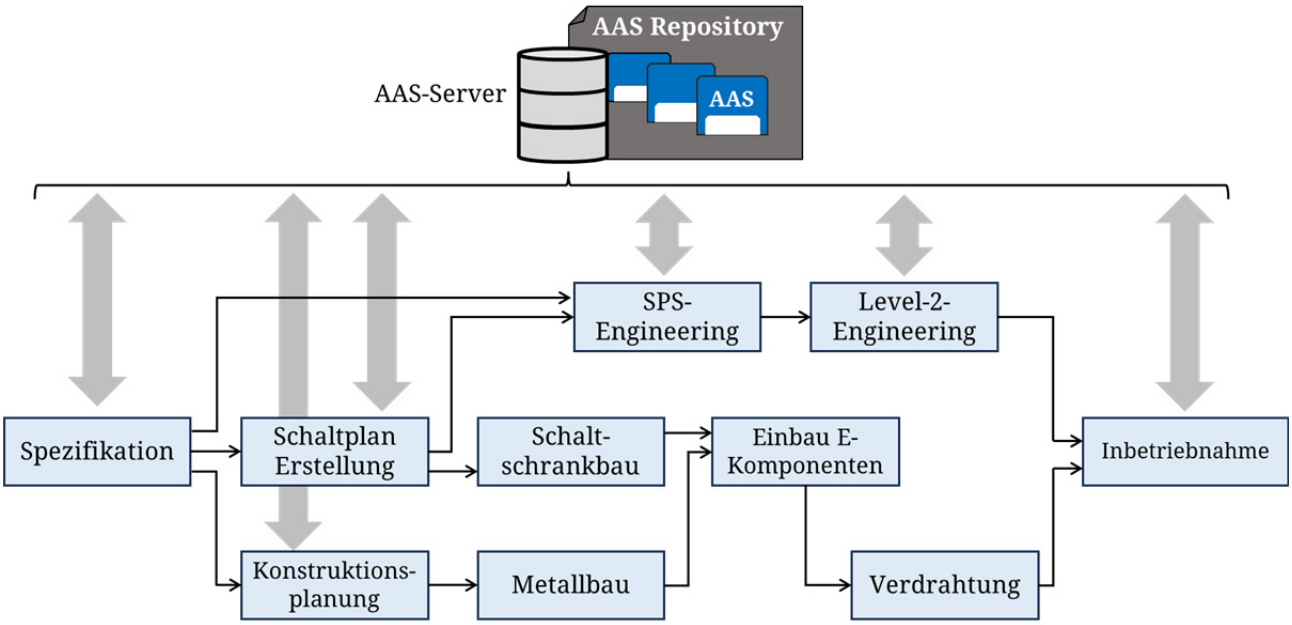

Bild 3: Interaktion verschiedener Gewerke/ Engineering-Werkzeuge mit der AAS

Da die für die Steuerungscodegenerierung relevanten Informationen in standardisierten Teilmodellen abgelegt sind, können Engineering-Werkzeuge diese Daten direkt aus den AASs beziehen. Dies umfasst sowohl die im Anlagenmodell enthaltenen Engineering-Daten (z. B. Strukturinformationen der Anlage, Schaltplan- oder Bestückungsdaten, Kommunikationszuordnungen) als auch die in Komponenten-AASs hinterlegten gerätespezifischen Informationen. Ein wesentlicher Vorteil liegt darin, dass diese Informationen nicht in separaten Modellen gepflegt oder manuell in Codegeneratoren eingegeben werden müssen, sondern bereits vorliegen, sobald die jeweiligen Komponenten instanziiert oder Engineering-Schritte abgeschlossen sind.

Durch den serverbasierten Zugriff auf AAS-Repositorys entfällt die für dateibasierte Informationsmodelle typische Notwendigkeit, Dateien auszutauschen, zu synchronisieren oder in unterschiedliche Werkzeuge zu importieren. Die Daten stehen allen beteiligten Engineering-Werkzeugen gleichzeitig und konsistent zur Verfügung. Dies ermöglicht es, dass mehrere Werkzeuge verschiedene Aspekte der AAS lesen oder aktualisieren können, ohne dass es zu Inkonsistenzen oder Versionskonflikten kommt.

Für die automatische Codegenerierung bedeutet dies, dass ein Codegenerator jederzeit auf die aktuellsten und vollständigen Informationen zugreifen kann. Beispielsweise können Konfigurationsdaten aus Komponenten-AASs unmittelbar genutzt werden, um Funktionsbausteine automatisiert zu parametrieren. Ebenso können Strukturinformationen aus der Anlagen-AAS verwendet werden, um

Variablenlisten, Ein-/Ausgangszuweisungen, Kommunikationszuordnungen oder feldgerätbezogene Initialisierungsbausteine automatisiert zu generieren. Durch die serverbasierte Bereitstellung entfällt der Aufwand, der bei traditionellen Ansätzen für das Einlesen, Pflegen oder Aktualisieren komplexer Modelle erforderlich ist.

AutomationML ist weiterhin ein wichtiges, etabliertes Format zur strukturierten Ablage von Engineering-Daten und kann innerhalb der AAS genutzt werden, beispielsweise zur Ablage kompletter Engineering-Dokumente oder zur Modellierung komplexer Strukturen. Die AAS ersetzt AutomationML also nicht, sondern baut vielmehr auf solchen etablierten Formaten auf und erweitert sie um ein serverbasiertes, verteiltes und lebenszyklusaktuelles Informationsmodell.

Der AAS-basierte Zugriff auf ein gemeinsames, lebenszyklusaktuelles Anlagenmodell bildet damit eine entscheidende Grundlage für die vollständige Mitigation des Informationsaufwands, der in bisherigen Ansätzen der automatischen Steuerungscodegenerierung erforderlich war. Codegeneratoren können Informationen automatisiert beziehen, ohne dass diese zuvor manuell strukturiert, modelliert oder in einzelnen Werkzeugen bereitgestellt werden müssen. Damit wird die in Hypothese 2 formulierte Annahme bestätigt, dass die AAS-Technologie einen effizienteren, kollaborativen und konsistenten Informationszugang für die automatische Steuerungscodegenerierung ermöglicht.

7 Fazit und Ausblick

Die im Beitrag vorgestellten Konzepte zeigen, dass die AAS-Technologie den Aufwand für die automatische Steuerungscodegenerierung deutlich reduzieren kann. Da im Laufe des Anlagenlebenszyklus viele relevante Informationen automatisch in den AASs entstehen und dort standardisiert abgelegt werden, stehen sie Engineering-Werkzeugen ohne zusätzlichen Pflegeaufwand zur Verfügung. Dadurch lassen sich wiederkehrende Aufgaben, wie das Parametrieren von Funktionsbausteinen oder das Erzeugen von Kommunikationszuordnungen, einfacher und zuverlässiger automatisieren.

Trotz dieser Vorteile ist eine vollständig automatische Generierung von Steuerungscode heute noch nicht erreichbar. Viele notwendige Informationen ergeben sich aus Kontextwissen, das in aktuellen Modellen nicht erfasst wird, etwa die genaue Verknüpfung einzelner Anlagenfunktionen und damit der (automatisch generierten Programmteile) oder projekt- und prozessspezifische Entscheidungen der Ingenieurinnen und Ingenieure.

Bild 4 stellt diese Problematik dar: Beispielhaft werden hier Variablenlisten, E/A-Konfigurationen, etc. aus in der AAS-Anlagenhierarchie abgelegten Schaltplänen generiert. Die Ablauflogik wird aus entsprechenden Beschreibungen abgeleitet, domänenspezifische Funktionen entstehen durch die Instanziierung von Funktionsbausteinen, die von den in der Anlage verbauten Komponenten über deren AASs, im Teilmodell „Software Nameplate“, bereitgestellt werden. Auch die Verriegelungslogik kann automatisiert aus R/I-Fließschemata erzeugt werden. Die Verknüpfung dieser einzelnen, automatisch generierten Programmteile zu einem vollständig lauffähigen Gesamtprogramm erfordert jedoch weiterhin menschliches Eingreifen.

In Zukunft könnten LLM-basierte Ansätze helfen, diese Lücke zu schließen. Solche Modelle sind grundsätzlich in der Lage, fehlende Zusammenhänge zu erkennen oder unvollständige Daten sinnvoll zu ergänzen. In Kombination mit den strukturierten AAS-Daten könnten sie somit einen wichtigen Baustein auf dem Weg zu stärker automatisierten und weitergehend autonomen Engineering-Prozessen darstellen.

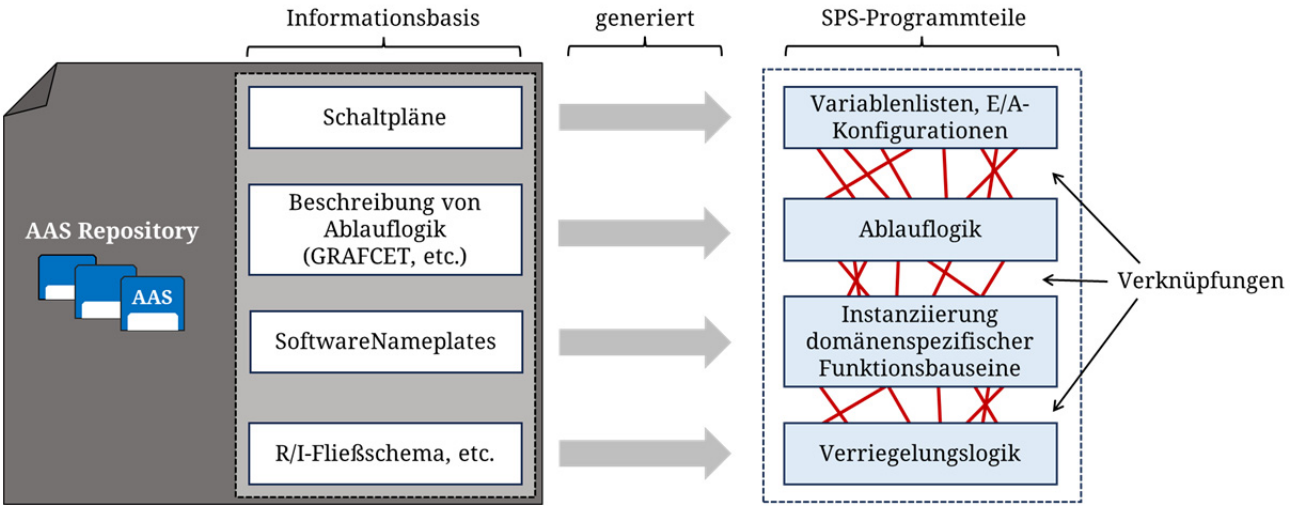

Bild 4: Verknüpfung von automatisch generierten SPS-Programmteilen

8 Literatur

[1] DIN EN IEC 62714-1: 2019-02 Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme - Automation markup language - Teil 1: Architektur und allgemeine Festlegungen.

[2] G. Frey, A. Schmidt: Automatische Erzeugung von SPS-Programmen aus Petrinetzen. In: Fachtagung Verteilte Automatisierung, 2000.

[3] J. Thieme, H.-M. Hanisch: Model-based generation of modular PLC code using IEC61131 function blocks. In: Proceedings of the 2002 IEEE International Symposium on Industrial Electronics, Vol. 1, S. 199–204, 08. - 11.07.2002.

[4] G. Music, D. Gradisar, D. Matko: IEC 61131-3 Compliant Control Code Generation from Discrete Event Models. In: Proceedings of the 13th Mediterranean Conference on Control and Automation, 27. - 29.06.2005.

[5] J. von Aspern: SPS-Softwareentwicklung mit Petrinetzen. Heidelberg: Hüthig Buch Verlag GmbH, 1993.

[6] F. Schumacher: Automatische Generierung von IEC 61131-3 Steuerungscode aus einer GRAFCET-Spezifikation. Hamburg, Helmut-Schmidt-Universität / Universität der Bundeswehr. Dissertation, 2013.
http://edoc.sub.unihamburg.de/hsu/volltexte/2013/3033/pdf/2013_Schumacher.pdf
[Abruf: 2018-01-24]

[7] DIN EN 60848: 2014-12 GRAFCET, Spezifikationssprache für Funktionspläne der Ablaufsteuerung.

[8] R. Julius, A. Fay: Konzept zur bidirektionalen Transformation zwischen GRAFCET Spezifikationen und IEC 61131-3 Steuerungscode. In: Entwurf komplexer Automatisierungssysteme, 2018.

[9] A. Würger: Automatisierte Generierung von Energiemanagementfunktionen auf der Basis des PROFINET-Energieprofils. Hamburg, Helmut-Schmidt-Universität / Universität der Bundeswehr. Dissertation, 2020. https://openhsu.ub.hsu-hh.de/handle/10.24405/9095 [Abruf: 2025-11-27]

[10] R. Drath, A. Fay, T. Schmidberger: Computer-aided design and implementation of interlock control code. In: Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, S. 2653–2658, 04. - 06.10.2006.

[11] K. Güttel, P. Weber, A. Fay: Automatic generation of PLC code beyond the nominal sequence. In: IEEE Conference on Emerging Technologies & Factory Automation, S. 1277–1284, 2008.

[12] K. Güttel: Konzept zur Generierung von Steuerungscode für Fertigungsanlagen unter Verwendung wissensbasierter Methoden. Hamburg, Helmut-Schmidt-Universität / Universität der Bundeswehr. Dissertation, 2012.

[13] L. Ren, H. Wang, J. Dong, H. Wang, S. Liu, Y. Laili, L. Zhang: MetaIndux-PLC: A Control Logic-Guided LLM for PLC code generation in industrial control systems. In: Applied Soft Computing Journal 184, 2025.

[14] IEC 63278-1:2023 Asset Administration Shell for industrial applications - Part 1: Asset Administration Shell structure.

[15] IEC 61406-1:2022 Identifizierungslink – Teil 1: Allgemeine Anforderungen.

[16] A. Würger: Digital Twin: Mehr Effizienz im Anlagen-Engineering. atp Edition (10/2024), S. 46–49.

[17] IDTA 02007-1-0 Nameplate for Software in Manufacturing, August 2023