Beschreibung von Prozessmodulen

Ein weiterer Schritt zur Umsetzung der NE 148

Mit dem Projekt Dima hat Wago Kontakttechnik zur Namur-Hauptsitzung 2014 einen Weg aufgezeigt, wie sich die in der NE 148 formulierten Anforderungen an die Integration von Prozessmodulen technisch umsetzen lassen. Basis des Lösungsansatzes ist die Beschreibung verschiedener Aspekte eines Moduls in einem proprietären Module Type Package (MTP). Die Kernelemente dieses MTP werden in diesem Beitrag offengelegt und zur Diskussion hinsichtlich der Verwendung in einer standardisierten Modulbeschreibung gestellt.

SCHLAGWÖRTER Dezentrale Intelligenz / Prozessmodul / Semantische Beschreibung / Plug-and-produce

Semantic Description of Process Modules – Towards an Open Implementation of the Requirements of the NE 148

With the Dima Project, Wago presented an approach at the Namur Annual General Meeting 2014 for the technical implementation of the requirements formulated in NE 148 for the integration of process modules. This approach is based on the description of various aspects of a module in a proprietary Module Type Package (MTP). The core elements of this MTP are presented here for consideration of their use in a standardised semantic description of process modules.

KEYWORDS decentralised intelligence / process module / semantic description / plug-and-produce

MICHAEL OBST, Technische Universität Dresden
THOMAS HOLM, Helmut-Schmidt-Universität Hamburg
LEON URBAS, Technische Universität Dresden
ALEXANDER FAY, Helmut-Schmidt-Universität Hamburg
SVEN KREFT, ULRICH HEMPEN, THOMAS ALBERS, Wago

Die NE 148 definiert Anforderungen an die Automation für die Umsetzung modularer Anlagenkonzepte [1, 2]. Diese Empfehlung ist bewusst technologieneutral gehalten, die Namur lässt die technische Lösung offen. Mit dem Projekt Dima hat Wago Kontakttechnik mit Unterstützung der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg und der Technischen Universität Dresden auf der Namur-Hauptsitzung 2014 in einem Demonstrator eine erste Umsetzung wesentlicher Anforderungen in dem Softwarewerkzeug e!Cockpit vorgestellt [4]. Die modulare Architektur verringert den Aufwand zur Integration eines Moduls in die Gesamtanlage, indem geeignete Aktivitäten in das Engineering des Modullieferanten verlagert werden [3]. Der Modullieferant projektiert das Modul und ist somit verantwortlich für die Einhaltung entsprechender Regularien und Normen. Ein Anlagenbetreiber führt das Integrationsengineering durch, bei dem die zuvor ausgewählten Module in das Prozessleitsystem (PLS) der Gesamtanlage integriert werden. Da das Modulengineering zeitlich vor dem Integrationsengineering stattfindet, müssen die Ergebnisse des Modulengineering in geeigneter Weise zwischengespeichert werden. Basis des Integrationsengineerings ist ein Informationsträger, der alle Modulinformation beinhaltet, die während der Integration eines Moduls benötigt wird. Die auf Basis der veränderten Engineeringprozesse erarbeitete Integrationsmethode ist in [3] ausführlich beschrieben. Dieser Beitrag widmet sich dem dabei genutzten Informationsträger, dem Module Type Package (MTP).

1. ASPEKTE DER MODULINTEGRATION

Bei modularen Anlagenarchitekturen muss die prozesstechnische Funktion der Gesamtanlage auf mehrere Module aufgeteilt werden. Wesentliche Anteile des Verhaltens der Anlage, also der Verfahrenstechnik, der Prozessführung und des Zeitverhaltens, sind somit direkt durch die Module und deren Aufbau bestimmt. Die Aspekte, die während des Integrationsengineerings von Modulen in ein übergeordnetes Prozessleitsystem benötigt werden, sind in der NE 148 [1] im Abschnitt „Datenaustausch und Informationsschnittstellen“ definiert. Darin werden neben den Strukturdaten, wie Prozessgrafiken, Verriegelungs-, Steuerungs-, und Regelungsstrukturen, Modulproxies- und Steuerungsfunktionen auch statische Daten zur Identifikation sowie dynamische Daten wie Prozesswerte, Sollwerte, Modi, Stati und Leistungsdaten genannt. Diese Daten sind notwendig, um das Prozessleitsystem zu befähigen, eine fehlerfreie Kommunikation zum Modul aufzubauen und die Modulfunktion in wunschgemäßer Ausführung abrufen zu können. Die soeben genannten Daten und Informationen ergeben aus einer funktionalen Sicht eine Klassifizierung, die auch unter der Bezeichnung „Tauchnitz‘sche Torte“ [5] bekannt geworden ist, siehe Bild 1.

Die identifizierten Funktionen entsprechen dort Stücken einer Torte und können dementsprechend zu einem gewissen Maß getrennt betrachtet werden. Für viele Anwendungsfälle ist nur ein Teil der dort genannten Funktionen Auswerten, Archivieren, Bedienen, Anzeigen, Überwachen und Alarmieren, Erfassen, Steuern, Regeln, Rezepte ausführen und Rezepte erstellen notwendig: Für eine zustandsbasierte Prozessführung, bei der die verfahrenstechnische Funktion eines Moduls in einem Dienst gekapselt wird und somit einer service-orientierten Architektur (SOA) entspricht, ist die Beschreibung der Dienste und deren Kommunikationsvariablen (SFC nach 61131, Batch nach 61512/S88, Konti nach S106) erforderlich. Um die Bedien- und Beobachtbarkeit zu realisieren, muss die abstrakte Beschreibung der Mensch-Maschine-Schnittstelle (Mimics, Trends, Key-Performance-Indikatoren) in das HMI-System eingelesen und ausgewertet werden. Sind Aspekte erforderlich, die über die Realisierung des Produktionsprozesses hinausgehen, müssen Alarmmanagement, Archivierung sowie Diagnose und Instandhaltung betrachtet werden. Weitere funktionale Teilmodelle sind denkbar, beispielsweise Simulationsmodelle für die virtuelle Wasserfahrt, Stücklisten der verbauten Geräte und Apparate oder Hard- und Softwareaspekte integrierte Fehlerausbrei-

tungsgraphen für eine modulübergreifende Fehleranalyse, siehe beispielsweise [6].

1.1 Bestehende Beschreibungsformate

Für die Feldgeräteintegration existieren bereits Beschreibungsmittel, mit denen ähnliche Aufgaben erfüllt werden können, wie sie bei der Integration von Modulen in ein Prozessleitsystem anfallen. So wurden zum Beispiel für die Package-Unit-Integration verschiedene Beschreibungsmittel aus dem Bereich der Feldgeräteintegration auf Eignung untersucht. Die Autoren von [7] analysieren standardisierte, industriell proprietäre und universitär entwickelte Beschreibungsmittel der Automatisierungstechnik, ob sie zur effizienten Abbildung der Information einer Package Unit geeignet sind. Dabei wurde zwischen Beschreibungsmitteln zur Beschreibung des System- oder Prozessverhaltens und der Bedienung unterschieden. Die als notwendig identifizierte Information wird auf die Sprachmittel und Eigenschaften von FDI abgebildet. Die Autoren kommen zu dem Schluss, dass wesentliche Aspekte der Integration nicht direkt in EDDL abgebildet werden können, und schlagen vor, diese Information entweder über Attachments oder als vorkompiliertes User Interface Plugin auszuliefern. Des Weiteren wird festgestellt, dass insbesondere standardisierte Beschreibungsmittel für Alarme und Archivierung fehlen. In [8] erweitern die Autoren die Analyse und zeigen auf, dass die Strukturmodelle von Bedienbildern und die Darstellung von Schrittketten in EDDL modellierbar sind. Die Autoren verlangen jedoch eine zwingende Erweiterung des Sprachumfangs von EDDL, um, wie auch von [9] in einer Anwendung für selbstbeschreibende Feldgeräte gefordert, semantische Verweise auf Merkmalsysteme wie eClass oder die in der Reihe IEC 60050 veröffentlichten internationalen elektrotechnischen Wörterbücher setzen zu können.

Die Autoren von [10] lösen sich von den Einschränkungen aktueller Strukturen der Automation und skizzieren eine Dienstearchitektur mit Industrie 4.0 (I40)-Komponenten. Für die Modulbeschreibung inspirierend ist vor allem die Werkzeugsammlung Sesame, die unter anderem eine umfangreiche Dienstespezifikationssprache bereitstellt [11]. Neben der Signatur eines Dienstes (Parameterlisten und deren Typen) können Vor- und Nachbedingungen für die Ausführung eines Dienstes beschrieben werden, um so für zustandsbehaftete Dienste zulässige Aufruffolgen definieren zu lassen. Anhand dieser Attribute wird festgestellt, ob in einer flexiblen I40-Anlage angefragte und angebotene Dienste kompatibel sind. Da der Aufbau des Zustandsmodells der IEC 61512/S88 gerade in dieser Hinsicht deutlich unterspezifiziert ist, ist noch zu prüfen, ob dieser Ansatz die notwendige Beschreibungsstärke für die Komposition von Prozessführungsstrategien für eine modulare Anlage besitzt. Die jüngst veröffentlichte NE 150 schließlich beschränkt sich bewusst zunächst auf den Austausch von I/O-Variablenlisten zwischen Engineeringwerkzeugen und Leitsystemen [12].

Abschließend lässt sich der Stand der Technik wie folgt zusammenfassen: Es existieren einige nicht oder nur lose verknüpfte Beschreibungssprachen und -mittel für ausgewählte Aspekte. Es gibt derzeit jedoch keine Lösung, die eine direkte Umsetzung der Idee der dezentralen Intelligenz für modulare Automation erlauben würde.

2. MODULE TYPE PACKAGE

Das im Folgenden vorgestellte MTP-Format beschränkt sich auf zwei wesentliche Aspekte der zustandsgeführten Prozessführung einer aus selbstständig automatisierten, mit dezentraler Intelligenz ausgestatteten Modulen aufgebauten prozesstechnischen Anlage. Dies entspricht der Typ-II-Moduldefinition der NE 148 [1].

2.1 Anforderungen

Ausgangspunkt der Definition der Anforderungen des MTP ist ein Geschäftsmodell, in dem Modulhersteller, wie in der NE 148 beschrieben, Module als eine feste und gegebenenfalls parametrierbare Einheit aus verfahrenstechnischer Funktion und Automatisierungstechnik anbieten. Die wesentlichen Engineeringschritte an der Schnittstelle von Verfahrenstechnik und Automatisierungstechnik finden beim Modulhersteller statt, ein Modulanwender soll ein solches Modul dann mit wenigen Schritten und ohne detaillierte Kenntnis des internen Aufbaus in eine modulare Anlage integrieren können. Daraus ergeben sich folgende Anforderungen an ein MTP:

Automatische Generierbarkeit: Ein MTP muss direkt und ohne manuelle Anreicherung aus den Engineeringdaten des Moduls generierbar sein. Dies ist eine wesentliche Voraussetzung für eine fehlerfreie und nachvollziehbare Parametrierung, Versionierung und Variantenverwaltung von Modulen. Bei der Modellierung ist daher darauf zu achten, dass keine manuellen Anreicherungen notwendig sind.

Unabhängigkeit von Beschreibungsaspekten: Eine weitgehende Unabhängigkeit der einzelnen Beschreibungsaspekte eines Moduls, sowohl methodisch als auch semantisch, stellt sicher, dass einzelne funktionale Aspekte (Tortenstücke) geordnet ausgetauscht und unabhängig voneinander entwickelt werden können. Dies unterstützt eine iterative, anforderungsgetriebene Entwicklung bestehender Aspekte des MTP und gewährleistet eine freie Erweiterbarkeit um neue Aspekte.

Trennung abstrakter und technischer Modelle: Die Informationsmodelle sollten so weit wie möglich technologieunabhängig spezifiziert werden können. Technologieabhängige Anteile eines Modells sind getrennt zu modellieren. Beispielsweise muss sich eine I/O-Variable unabhängig von den Einschränkungen der Kommunikationstechnologie identifizieren und benennen lassen. Die Abbildung auf die kommunikationsmittelspezifische Adressierung – beispielsweise Coil/Index

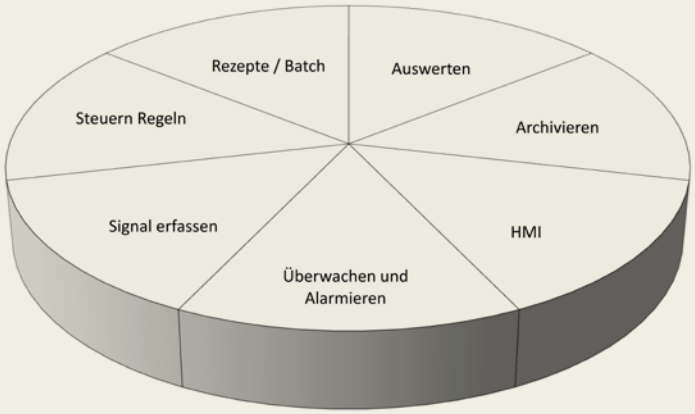


BILD 1: Darstellung der funktionalen PLS-Aspekte in der Tauchnitz´schen Torte [4]

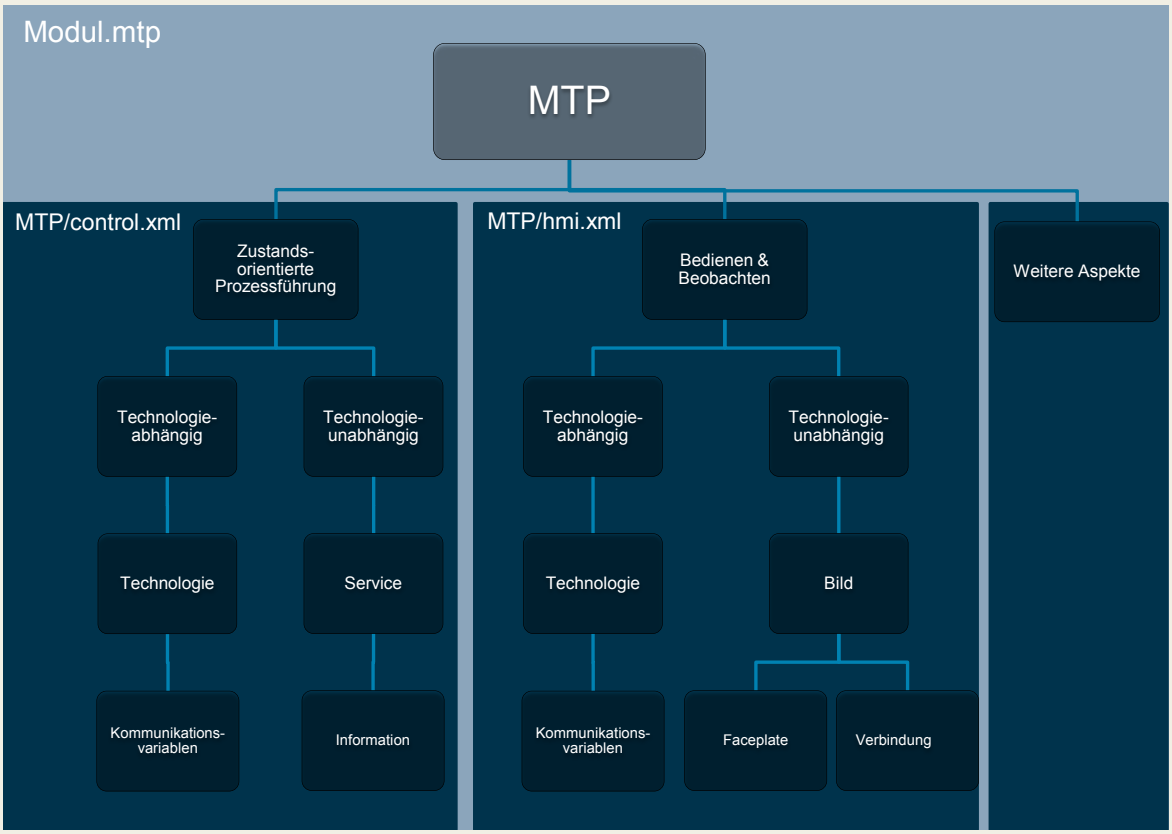


BILD 2: Inhaltliche Strukturierung des MTP

```
<pconnection type="OPCDA" server="CodeSysDA">
    <!-- Each pvalue defines a variable for Communication between hmi and module-->
    <pvalue pvid="c5232ef6-6cd6-4387-b7e1-fec9f5a5020c" name="Tempering.Temperier_Regel_D" description="">
        <!-- In this case (OPC DA) this is the OPC TAG-->
        <adress>_Temperierer.Application.PLC_PRG.Tempering.Temperier_Regel_D</adress>
        <accesstyp>READWRITE</accesstyp>
        <datatype>REAL</datatype>
    </pvalue>
```

BILD 3: Beschreibung der technologieabhängigen Kommunikationsvariablen am Beispiel von OPC DA

bei Modbus, Slot/Index bei Profibus oder server/tagname bei OPC – sind in einem technologieabhängigen Anteil zu kapseln.

Automatische Prüfbarkeit und Lesbarkeit: Das integrierende System muss den MTP einlesen und die notwendigen Objekte im eigenen Datenhaushalt automatisch generieren. Das MTP-Format muss daher einfach auf syntaktische Richtigkeit und Vollständigkeit geprüft werden können. Auch beim Prüfen und Einlesen sind Datentypen, Strukturen und Kennzeichnungssysteme, die einen händischen Eingriff erfordern, kontraproduktiv. Eine gegebenenfalls notwendige Auflösung von Vieldeutigkeiten (beispielsweise bei der Konfliktresolution in einem multi-kriteriellen Optimierungsschritt wie der Layoutanpassung von HMI an andere Formfaktoren) muss wiederholbar parametrierbar sein.

Die konkreten Daten eines MTP beschreiben immer eine einzige Modulinstanz. Über eine Serien- und Versionsnummer wird die eindeutige Identifikation gewährleistet. Ein MTP bildet immer den, gegebenenfalls auch vorkonfigurierten, Aufbau und die Fähigkeiten des Moduls ab, die durch das Automatisierungsprojekt des Moduls vorgegeben werden. Spätere Änderungen, beispielsweise der Kommunikationsparameter (IP-Adresse), müssen in dem zu integrierenden Automatisierungssystem über eine Parametrierung eingepflegt werden. Bei einem Abkoppeln des Moduls muss die Eindeutigkeit zwischen MTP und Modul gewährleistet werden. Hier ist es vorstellbar, dass durch den Vorgang des Abkoppelns das Modul in den Auslieferungszustand zurückgesetzt wird oder eine entsprechende Änderung im MTP generiert wird, die zum Beispiel die veränderte Lebenszyklusinformation beinhaltet.

2.2 Strukturierung

Zur Unterstützung der Versionierbarkeit und Verteilung eines MTP ist dieser als Modellcontainer mit einer hierarchischen Ordnerstruktur angelegt. Hierfür können beliebige Archivformate eingesetzt werden. In dem in [4] vorgestellten Demonstrator kommt ein ZIP-Archiv zum Einsatz. Zum einen sind für viele Sprachen und Systeme open-source-Implementierungen von Funktionssammlungen verfügbar, zum anderen sind Funktionen zum Umgang mit ZIP-Archiven in nahezu allen gängigen Betriebssystemen verfügbar. Im Folgenden werden die Inhalte dieses Modellcontainers beschrieben.

Manifest: An der MTP-Wurzel ist eine Manifest-Datei angelegt. Diese XML-Datei enthält die für den MTP relevanten Meta-Daten. Diese geben Auskunft über die Version der zugrunde gelegten MTP-Spezifikation, über die eindeutige Seriennummer des MTP und beschreiben die in diesem MTP enthaltenen funktionalen Aspekte. Dies beschleunigt die Indizierung und Suche eines prinzipiell geeigneten MTP, da hierzu zunächst nur die Manifest-Daten ausgewertet werden müssen. Die in der Manifest-Datei abgelegte Information zu den Aspekten (Tortenstücke) beinhaltet Typ, Pfad zur entsprechenden Datei, Version und ein Flag, das anzeigt, ob der enthaltene Aspekt aktiv ist. Dies erlaubt beispielsweise bei komplexen Produktfamilien, den Modulvarianten Beschreibungsvarianten zuzuweisen und die relevanten Teilmodelle zu- oder abzuschalten (aktiv beziehungsweise inaktiv). Weiterhin enthält das Manifest vordefinierte Einträge für häufig genutzte Dateiarten, beispielsweise Bilder (*images*), Zertifikate (*certificates*), Hilfedateien (*helpfiles*) und Handbücher (*manuals*). Eine weitere wichtige Informationsklasse sind Angaben zur Kommunikation mit der Recheneinheit des Moduls (SPS oder Embedded Controller).

Integrationsaspekte: Jeder eigenständige Integrationsaspekt wird in einer eigenen Datei abgelegt, auf die die Manifest-Datei verweist. Durch diese strikte Trennung der Einzelmodelle ist eine einfache Erweiterung gegeben und das Hinzufügen neuer Integrationsaspekte möglich. Bild 2 stellt die hierarchische Strukturierung des MTP (Modul.mtp) dar. Im Projekt Dima [4] wurden zwei Integrationsaspekte untersucht, das Bedienen und Beobachten (hmi.xml) und die zustandsbasierte Prozessführung (control.xml); diese werden im Folgenden detailliert beschrieben. Die weitere Aufteilung der einzelnen XML-Dateien in einen technologieabhängigen und technologieunabhängigen Teil erfolgt durch die Strukturierungsmöglichkeiten der XML.

2.3 Technologieunabhängige und -spezifische Teilmodelle

Information, also die Verbindung von Struktur (Syntax) mit der Bedeutung der Strukturentitäten (Semantik), kann unabhängig von der Abbildung dieser Information durch Sprach- und Strukturmittel einer Kommunikationstechnologie, wie zum Beispiel OPC DA, dargestellt werden. Auch innerhalb der Dateien für die jeweiligen Beschreibungsaspekte wird zwischen der Beschreibung der Variablen und Funktion des Moduls und deren Abbildung auf Kommunikationsvariablen einer Kommunikationstechnologie getrennt. Durch diese Trennung kann der Modulhersteller mehrere Kommunikationskanäle vorsehen. Weiterhin ist es möglich, die inhaltliche Beschreibung des Moduls getrennt von möglichen Kommunikationstechnologien zu entwickeln.

Für die Beschreibung der Prozessführung und die Beschreibung der Funktionen des Bedienens und Beobachtens wurde der gleiche Ansatz zur Beschreibung der Kommunikation gewählt. Im weiteren Verlauf des Beitrags werden die implementierten Anteile der technologieunabhängigen Beschreibung für die Aspekte Prozessführung und Bedienen und Beobachten näher erläutert.

2.4 Technologieabhängige Beschreibung der Kommunikation

Die Beschreibung der realen Kommunikation zwischen dem PLS und dem Modul muss technologieabhängig erfolgen.

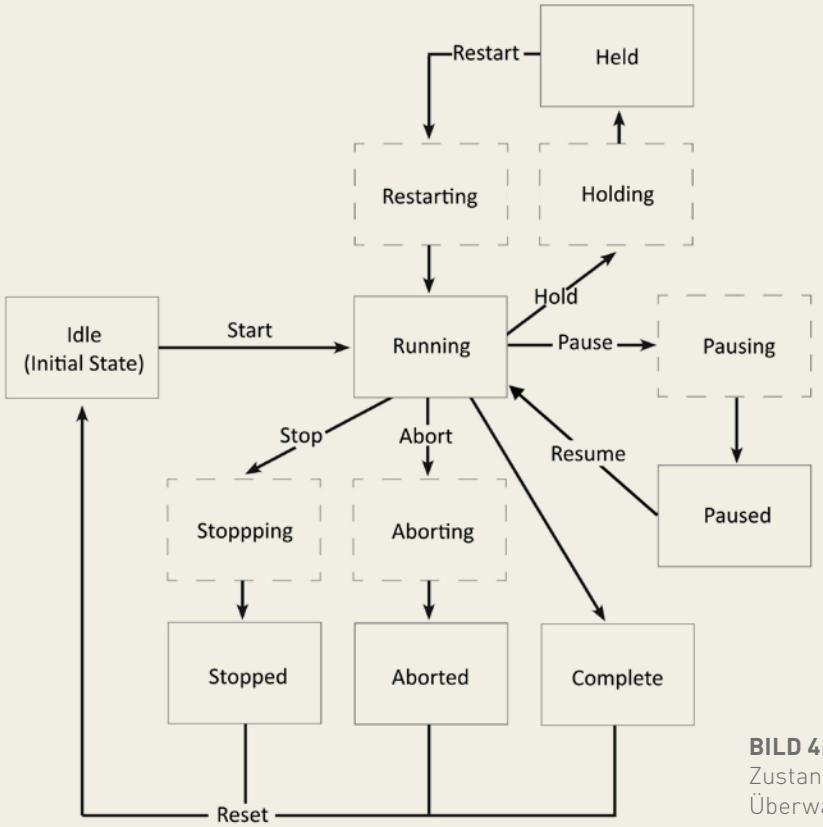


BILD 4: Darstellung der Zustände und Zustandsübergänge zur Steuerung und Überwachung der Dienste

```
<modul id="" name="Plant_Module_1" ref="">
    <services count="2">
        <service id="72004906-ce30-49f5-801f-258d73a27347" name="Tempering" ref="" >
            <ServiceInterfaceInfos>
                <ServiceInterfaceInfo name="Tempering_Com._F" pvid="10cc5198-d910-4858-bc26-79c47bb749fc" ref=""/>
                <ServiceInterfaceInfo name="Tempering_Com._P" pvid="afb8eaa1-abde-458c-b2bf-f8705b4160b1" ref=""/>
                <ServiceInterfaceInfo name="TemperiererVorlauftemp" pvid="0a87e91f-9c5f-4543-805f-c1afbdac34db" ref=""/>
            </ServiceInterfaceInfos>
        </service>
        <service id="342b4ee7-6c6e-4a88-b073-f3fabbfd1f54" name="Refilling" ref="" >
            <ServiceInterfaceInfos>
                <ServiceInterfaceInfo name="Refilling_Com._F" pvid="846fb728-7fac-4fd8-ac46-56ec5d2c7cab" ref=""/>
                <ServiceInterfaceInfo name="Refilling_Com._P" pvid="22420bdf-c13e-4d1f-8db9-a4d448996475" ref=""/>
            </ServiceInterfaceInfos>
        </service>
    </services>
</modul>
```

BILD 5: XML-Darstellung des technologieunabhängigen Anteils für die Prozessführung des Moduls

Für jede Kommunikationstechnologie, die das Modul unterstützt, wird ein neues Element *pconnection* eingefügt. Dieser Teil beinhaltet die Definition der Kommunikationsparameter sowie die Beschreibung der einzelnen Variablen *pvalue* spezifisch für die Kommunikationstechnologie. Die logische Verbindung zwischen der technologieunabhängigen und technologiebezogenen Variablen wird über eine im MTP eindeutige ID *pvid* hergestellt. Bild 3 stellt eine entsprechende Variable unter Nutzung der OPC-DA-Technologie dar.

2.5 Zustandsbasierte Prozessführung

Die prozesstechnische Funktion wird bei modularen Anlagenarchitekturen in den Modulen gekapselt. Die gewünschte Wandlungsfähigkeit dieser Produktionsanlagen bedingt, auch die Kommunikation in geeigneter Weise zu kapseln. Die Prozessführung dieser Module erfolgt in der Dima-Methodik aus diesem Grund über eine Diensteschnittstelle. Diese folgt den Grundsätzen, die durch die Oasis-Referenz-Architektur

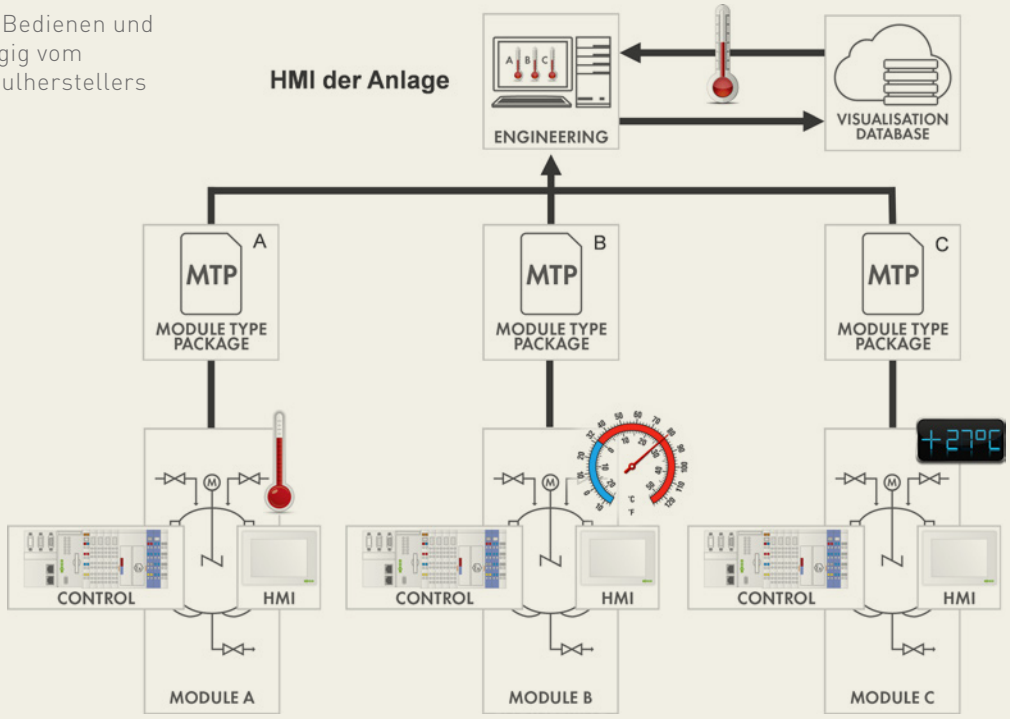


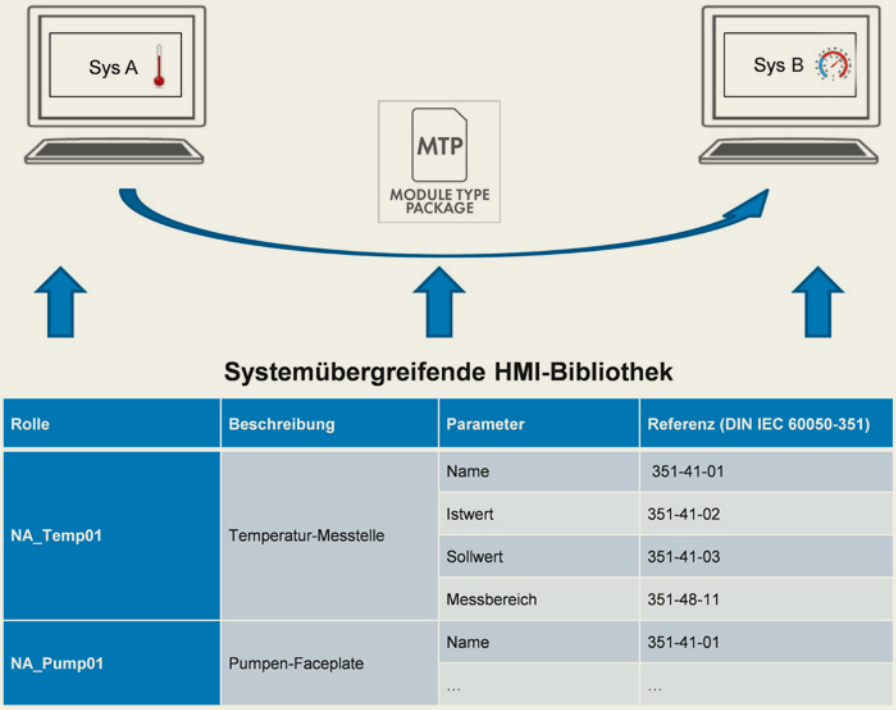
BILD 6: Einheitliches Bedienen und Beobachten unabhängig vom Quellsystem des Modulherstellers

Foundation [13] formuliert wurden. Die Dienste werden dabei über zuvor definierte Zustände und Zustandsübergänge gesteuert und überwacht, siehe Bild 4. Das heißt, die Kommunikation zwischen PLS und Modul bedingt Kommunikationsvariablen, die die Aufforderung zur Erreichung eines der Zustände, beziehungsweise die das Ergebnis über das Erreichen eines Zustandes abbilden. Im Rahmen der prototypischen Umsetzung des Projektes Dima wurde als Methodik zum Austausch dieser Information das Phase Logical Interface (PLI) [14] genutzt. Die Kapselung der prozesstechnischen Funktion in einem Dienst muss allerdings anpassbar sein, um die erforderlichen produkt- und prozessspezifischen Gegebenheiten zu ermöglichen. Aus diesem Grund sind Parameter zur Anpassung des Dienstes vorzusehen. Mit ihnen können Sollwerte vorgegeben und aktuelle Istwerte abgefragt werden. Die Parameter müssen dazu sowohl Grenzwerte, also die Spanne zwischen Minimum und Maximum, in der ein gefahrloser und zerstörungsfreier Betrieb möglich ist, beinhalten, als auch deren Einheiten. Dazu sind ebenfalls Kommunikationsvariablen vorzusehen.

Der technologieunabhängige Anteil der Beschreibungsdatei für die zustandsbasierte Prozessführung enthält Information zu den Diensten und den zugeordneten Kommunikationsvariablen für die Zustandsanforderungen und -erreichung sowie die Variablen, mit denen die Parametrierung erfolgen kann. Dem Element des Moduls (*modul*) wird dazu ein weiteres Kindelement mit Attribut zur Anzahl der Dienste eingefügt (*services*). Das Element services enthält die Dienste (*service*) mit den entsprechenden Variablen. Jede Variable wird durch ein Element *ServiceInterfaceInfo* repräsentiert. Um Parametervariablen und Variablen zur Zustandskommunikation unterscheiden zu können, wird ein entsprechender Eintrag im Attribut *ref* genutzt. Hier sind Eintragungen mit Verweis auf zum Beispiel eClass-Elemente [15] oder auch Ontologien wie QUDT [16] denkbar. Bild 5 stellt den technologieunabhängigen Anteil des Moduls *Plant_Module_1* dar. Es verfügt über zwei Dienste mit jeweils zwei Variablen zur Zustandskommunikation (ref-Eintrag nicht abgebildet). Der Dienst *Tempering* beinhaltet des Weiteren eine Parametervariable (ref-Eintrag nicht abgebildet).

2.6 Bedienen und Beobachten

Die VDI-Richtlinie 3699 [17, S.9] beschreibt die Prozessführung als „eine Aufgabe des Operators mit dem Ziel, den bestimmungsgemäßen Betrieb einer verfahrenstechnischen Anlage wirtschaftlich und umweltverträglich durchzuführen. Ergänzend dient dem sicheren Betrieb das Schutzsystem, das der Operator nicht beeinflussen kann.“ Die nutzerfreundliche Gestaltung der Mensch-Maschine-Schnittstellen der Prozessleitsysteme (PLS) stellt somit einen wichtigen Faktor für den wirtschaftlichen Betrieb von Anlagen dar. Durch die Heterogenität der Engineeringwerkzeuge bei der Erstel-



Rolle	Beschreibung	Parameter	Referenz (DIN IEC 60050-351)
NA_Temp01	Temperatur-Messstelle	Name	351-41-01
		Istwert	351-41-02
		Sollwert	351-41-03
		Messbereich	351-48-11
NA_Pump01	Pumpen-Faceplate	Name	351-41-01
		...	...

BILD 7: Rollenbasiertes Bibliothekskonzept zur HMI Integration

lung der modulspezifischen Bedienbilder ist es Ziel des MTP, diese Bedienbilder in einer neutralen Beschreibungsform abzubilden. Weiterhin muss den spezifischen Darstellungsmöglichkeiten der Bedien- und Beobachtungsysteme sowie projektspezifischen Anforderungen an die Darstellung einzelner Elemente eines Bedienbildes Rechnung getragen werden. Ebenfalls muss bei der Integration von Modulen unterschiedlicher Hersteller eine einheitliche Darstellung der Bedienelemente über die Modulgrenzen hinweg gewährleistet sein, siehe Bild 6. Daher setzt das Dima-Konzept auf eine rollenbasierte Beschreibung der einzelnen Bedienelemente, siehe Bild 7.

Sowohl auf Seiten der Bedienbildererstellung beim Modulhersteller (Sys A) als auch beim Bedien- und Beobachtungsystem des Modulbetreibers (Sys B) müssen Bibliotheken zum Einsatz kommen, die um eine semantische Bedeutung der einzelnen Bedienelemente ergänzt wurden (Bild 7). Dazu muss in einem Harmonisierungsprozess ein minimales Set an Elementen identifiziert werden, die zum Bedienen mehrerer Module notwendig sind. Dabei bedarf es einzig der Abstimmung, welche Elemente (zum Beispiel Ventil, Antrieb, Messstelle) mit welcher Information (beispielsweise Sollwert, Istwert) es zu übertragen gilt. Keineswegs müssen die grafischen Darstellungen des Bedienelementes oder das Interaktionsverhalten harmonisiert werden. Dies ist und bleibt Differenzierungsmerkmal der integrierenden Systeme.

Abstrakte Beschreibung von Bedienbildern

Bedienbilder der Prozessindustrie bilden implizit und explizit Strukturinformation des geführten Prozesses und der Automation ab. Die einzelnen aktiven visuellen Elemente, die Faceplates, sind sichtbar über Rohrleitungen und Wirklinien miteinander verknüpft. Die aus der technologischen Kompositionshierarchie ableitbaren ist-Teil-von-Beziehungen der Elemente werden üblicherweise indirekt durch grafische Mittel des Layouts wie örtliche Nähe und regelmäßige Anordnung dargestellt. Die direkten Strukturmittel sind häufig so stark ausgeprägt, dass die Autoren von [18] einen Ansatz vorstellen, wie die HMI in Brownfieldprojekten, in denen keine anderweitigen Informationsquellen zur Verfügung stehen, ausgewertet werden können, um die Anlagentopologie zu rekonstruieren. Die konkrete Ausprägung des Layouts durch 2-dimensionale x,y-Koordinaten ist jedoch sehr stark abhängig von der Implementierungsplattform. Um die geforderte Technologieunabhängigkeit zu wahren, sollen die Beziehungen zwischen den Elementen eines Bedienbilds in dem MTP mathematisch-abstrakt als Graph, bestehend aus Knoten und Kanten, beschrieben werden. Die x,y-Koordinaten der Faceplates im Modul-Engineeringsystem werden als Attribute an den Knoten mitgeführt und können vom Integrations-Engineeringsystem als Layouthinweise genutzt werden. Im einfachsten Fall, wenn die Bildschirmgröße und Faceplategröße von Quell-

```
<graph id="Visu_Temp" name="Visu_Temp" w="1920" h="1200">

    <node id="2a175c26-8071-4c66-86cf-a26c875dae44" tagname="L001" x="446" y="594" w="371" h="469" role="NA
    <node id="7369b780-9b9b-4cfb-9b29-5a95e56e1473" tagname="P001" x="937" y="1025" w="95" h="145" role="NA
    <node id="dd174dd1-c603-4260-a77d-fe80e8e34977" tagname="X001" x="914" y="354" w="106" h="153" role="NA

    <edge id="fd6f47d1-7491-4aa6-9446-8142633c0c22" name="Pipe_01" role="NA_Pipe01" source="2a175c26-8071-4
    <edge id="cde7ce82-e99e-4324-b3bf-46b578f1ec52" name="Pipe_02" role="NA_Pipe01" source="fd6f47d1-7491-4
    <edge id="2ec85233-29b7-49c1-8e02-49817e791f27" name="Pipe_03" role="NA_Pipe01" source="7369b780-9b9b-4
    <edge id="c57dbdb6-9f2d-4b6f-a8b2-6494d31c8759" name="NA_Pipe01_09" role="NA_Pipe01" source="dd174dd1-c
</graph>
```

BILD 8: Beschreibungen der Anlagentopologie eines Bedienbildes in GraphML

```
<node id="7369b780-9b9b-4cfb-9b29-5a95e56e1473" tagname="P001" x="937" y="1025"
          w="95" h="145" role="NA_Pump01" ref="">
    <data>
        <variable informationtag="NA_351-41-02" pvid="5de8891b-4319-4d29-84c4-6281588267e1"
        ref="http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=351-41-02"/>
    </data>
</node>
```

BILD 9: Beschreibungen der statischen Layoutparameter, der dynamischen Elemente und der Semantik eines Faceplates mit GraphML

und Zielsystem weitgehend übereinstimmen, lassen sich diese Layouthinweise direkt übernehmen.

Abbildung mit GraphML

Aufgrund der guten Einbindung in Open-Source-Editoren für die Validierung der Beschreibung und der guten Unterstützung durch graphentheoretische Funktions- und Algorithmensammlungen fiel im Dima-Projekt die Wahl auf das XML-basierte Beschreibungsmittel GraphML [19]. Weitere in der Fachliteratur beschriebene Lösungsansätze zur Automatisierung der Bedienbildgenerierung nutzen HTML5 [24], SVG [20] oder proprietäre HMI-Formate [21-23]. Diese Formate benötigen entweder umfangreiche MDA-Rahmenwerke [21-23], sind für die Prozessautomation nur bedingt geeignet [22-24] oder lassen sich nur mit speziellen Ergänzungen mit den für ein semantisches Layout notwendigen Attributen erweitern [20, 24].

Jedes Bedienbild eines Moduls wird als einzelner Graph mit dem in der GraphML-Spezifikation definierten Element vom Typ *graph* beschrieben, siehe Bild 8. Die *graph*-Elemente der Bedienbilder sind Kindelemente des Sammelelements *pictures*, das sich direkt unterhalb des Wurzelelements *HMI* befindet. Parallel zu *pictures* liegt das Element *pconnections*, das die technologieabhängigen Kommunikationskanäle der in den Bedienbildern darzustellenden und zu manipulierenden Variablen des Moduls beschreibt.

Jedes Faceplate wird durch das in GraphML spezifizierte XML-Element *node* repräsentiert, jede Verbindung zwischen Faceplates (Rohrleitung, Wirklinie, Hierarchie) wird durch das Kanten-Element *edge* beschrieben. Die die Faceplates repräsentierenden Knoten-Elemente werden durch folgende skalare Attribute näher bestimmt:

- **id:** Diese für den Graphen ein-eindeutige, rechnererzeugte Identifikation erlaubt Querverweise, beispielsweise als Start- und Endpunkt einer Kante. Dieses Attribut ist in GraphML definiert und notwendig.
- **tagname:** Eine kontextbezogen eindeutige Kennung für den Bediener. MTP-spezifische Erweiterung von GraphML, notwendig
- **x,y,w,h:** Koordinaten und Abmaße des Faceplates im Quellengineeringsystem als Hinweis für das Layoutsystem des integrierenden Systems. MTP-spezifische Erweiterung von GraphML, optional
- **role:** Über eine eindeutige und weitgehend generische Rollenbezeichnung werden in den Engineeringwerkzeugen die zu plazierenden Faceplates ausgewählt. MTP-spezifische Erweiterung von GraphML, notwendig
- **ref:** semantischer Verweis auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass und IEV zur weiteren Spezialisierung der in dem Attribut *role* definierten Rolle. MTP-spezifische Erweiterung von GraphML, optional

Für die Beschreibung nicht-skalarer Elemente von Knoten sieht GraphML ein *data*-Element vor, Bild 9. In diesem Block werden die für den Knoten relevanten Prozessvariablen, wie Istwert, Sollwerte, Grenzwerte, als *variable*-Elemente aufgelistet. Jedes *variable*-Element muss mit folgenden Attributen ausgezeichnet werden:

- **informationtag:** Rolleninformation entsprechend der Bibliothek
- **pvid:** Referenz auf die Kommunikations-Variable in dem *pconnection*-Element
- **ref:** semantischer Verweis auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass und IEV

Die mit *edge*-Elementen beschriebene konkrete oder abstrakte Verknüpfung zwischen zwei Knoten Rohrleitung, Wirklinie oder ist-Teil-von-Beziehung werden durch folgende Attribute näher spezifiziert:

- **target, source:** Definition der Anfangs- und Endknoten einer Verbindung. Definiert in GraphML, notwendig
- **edgepath:** Verlauf der Kante im Quellsystem als Hinweis für das Layout im Integrationssystem. MTP-spezifische Erweiterung von GraphML, optional
- **edgeio:** Koordinaten von Anfang und Ende als Hinweis für das Layout im Integrationssystem. MTP-spezifische Erweiterung von GraphML, optional
- **ref:** Semantischer Verweis auf ein standardisiertes Merkmal- oder Taxonomiesystem wie eClass, IEV oder RDFS zur Bestimmung der Bedeutung der Kante. Wird keine Referenz angegeben, soll das integrierende System diese Kante wie eine unspezifizierte Rohrleitung behandeln. MTP-spezifische Erweiterung von GraphML, optional

Wie das *node*-Element kann das *edge*-Element mit beliebigen weiteren, anwendungsspezfischen Attributen und einem *data*-Element erweitert werden. Relevante Erweiterungen sind Hinweise zur Auszeichnung von statischen Rohrleitungsklassen oder Variablen zur Visualisierung von dynamischer Routing-Information. Dabei ist zu beachten, dass bereits mit den wenigen vorgestellten Elementen ein Großteil der statischen und dynamischen Elemente verschiedener Arten von Bedienbildern beschrieben und im Look-and-feel des Integrationssystems instanziiert werden kann.

Ein kritischer Punkt für die Akzeptanz dieses Ansatzes ist sicherlich der Aufwand für die (einmalige) Implementierung in den Engieneeringsystemen und der zusätzliche Aufwand für die Modulingenieure zur automatischen Generierung der Instanzmodelle. Im Projekt Dima wurden Exportschnittstellen für CodeSys als Modulengineeringwerkzeug und Importschnittstellen für PCS 7 als Integrationsengineeringwerkzeug prototypisch implementiert. Mit den verfügbaren kommerziellen und open-source-Libraries für XML und GraphML waren Export und Import schnell realisiert, und das Entwicklungsteam konnte sich auf die werkzeugspezifischen Datenstrukturen und Application Programming Interfaces konzentrieren. Da die Datenstrukturen des Modul-HMI-Engineering-Systems nicht verändert werden sollten, wurde die für den Export notwendige Rolleninformation als Präfix des Symbolnamen angegeben, beispielsweise *NA_Pump01%P001* für das in Bild 9 aufgeführte Faceplate einer Pumpe.

Die automatische Erstellung der Kanten der Hierarchie-Relationen zwischen den Elementen erfordert größere Eingriffe in das Modul-HMI-Engineeringsystem, wenn diese nicht automatisch aus dem Kennzeichnungssystem rekonstruiert oder über Objektverweise auf eine technologische Hierarchie entnommen werden können. Da diese Relationen erst dann benötigt werden, wenn sich Bildschirme und Faceplates der Quell- und Zielsysteme bezüglich der Formfaktoren deutlich unterscheiden und auf dem Zielsystem ein automatischer Layoutlauf vorgenommen werden muss, wurden diese Beziehungen als Elemente des HMI-Graphen im Sinne einer 20:80-Lösung als optional klassifiziert.

AUSBLICK

Der vorgestellte MTP stellt einen aktuellen Arbeitsstand des Projektes Dima dar. Im Rahmen dieses Betrags wird der MTP nun der Fachgemeinde präsentiert und zur Diskussion gestellt. Dies soll eine breite und vor allem herstellerübergreifende Diskussion über Weiterentwicklung und Einsatz des MTP ermöglichen. Dazu sind zwei weitgehend unabhängige Standardisierungsaspekte zu verfolgen – zum einen die Festlegung der notwendigen Inhalte, zum anderen die Definition von Symbolnamen, Strukturen und Distributionsformaten. Der vorgelegte MTP ist als Container definiert, der einzelne Modelldateien enthält. Dieser Ansatz ermöglicht es, die Teilaspekte eines MTP aus verschiedenen, unabhängigen Werkzeugketten ohne Zusatzaufwand und unter Rückgriff auf bekannte Open-Source-Werkzeuge, wie beispielsweise *make* oder *zip*, zu erzeugen. Zusätzlich in der Automation weit verbreitete Kandidaten für das Distributionsformat sind AutomationML oder FDI. Beide Distributionsformate geben weder Bedeutung von Inhalten noch die Beschreibungsebene eines Moduls vor, sie enthalten jedoch Vorgaben und Rahmen für Syntax und Strukturierung der Informationsmodelle.

Weiterhin wurde der MTP dem Namur-AK 1.12 und dem ZVEI-AK Modularisierung vorgestellt, um die nächsten notwendigen Schritte hinsichtlich der Standardisierung zu starten. Die folgende Auswahl gibt einen kompakten Einblick über mögliche nächste Schritte beziehungsweise zu klärende Fragestellungen:

- Erweiterung der zu harmonisierenden Inhalte der Aspekte entsprechend der Tauchnitz´schen Torte: zum Beispiel Diagnose, Alarm, Archiv
- Der Dima-Ansatz setzt ein Modul mit einer frei programmierbaren Steuerungskomponente voraus. Wie können Module mit einer Remote I/O mit dem MTP beschrieben werden? Nicht alle Aspekte der Integration, zum Beispiel die Prozessführung über Services, können dann umgesetzt werden.
- Der MTP beschreibt genau eine Instanz eines Moduls. Wie verhält sich das Automatisierungssystem beim Abkoppeln eines Moduls? Wird das Modul in seinen Auslieferungszustand versetzt? Ändert des AT-System den MTP (beispielsweise durch Einstellen der neuen IP Adresse) oder wird dieser mit einer Historie versehen? Wer ist dann für die Pflege des MTP verantwortlich?

- Wie erfolgt die Diagnose der einzelnen intelligenten Feldgeräte des Moduls über die Grenzen des Moduls hinweg?
- Wie können, bei Einsatz von Batch-Werkzeugen zur zustandsbasierten Prozessführung, die meist werkzeugspezifischen Phasen-Logiken aufeinander abgebildet werden?

Die Aufstellung zeigt die vielfältigen Aufgaben bei der Standardisierung der Modulbeschreibung für eine einfache, schnelle und flexible Integration von Modulen in ein Automatisierungssystem. Im Rahmen der Tätigkeiten der entsprechenden Namur-AK und des ZVEI-AK Modularisierung soll nun die standardisierte Beschreibung von Prozessmodulen vorangebracht werden, um der Umsetzung der NE 148 einen weiteren Schritt näher zu kommen.

MANUSKRIPTEINGANG
19.01.2015
Im Peer-Review-Verfahren begutachtet

REFERENZEN

[1] NE 148: Anforderungen für Modularisierung verfahrenstechnischer Anlagen. Namur, 2013
[2] Urbas, L.; Bleuel, St.; Jäger, T.; Schmitz, St.; Evertz, L.; Nekolla, T.: Automatisierung von Prozessmodulen. Von Package-Unit-Integration zu modularen Anlagen. atp edition – Automatisierungstechnische Praxis 54(1-2), S.44-53, 2012
[3] Holm, Th.; Fay, A.: Aufwandsbetrachtung im Engineering modularer Prozessanlagen; Prozess-, Apparate- und Anlagentechnik (PAAT), 17.-18.11.2014, Lüneburg
[4] Holm, Th.; Obst, M.; Fay, A.; Urbas, L.; Albers, Th.; Kreft, S.; Hempen, U.: Dezentrale Intelligenz für modulare Automation. atp edition – Automatisierungstechnische Praxis 56(11), S.34-43, 2014
[5] Tauchnitz, Th.: Die „neuen Prozeßleitsysteme" – wohin geht die Reise? atp – Automatisierungstechnische Praxis 38(11), S. 12-23, 1996
[6] Morozov, A.; Janschek, K.: Dual Graph Error Propagation Model for Mechatronic System Analysis. In: Proceedings of the 18th IFAC World Congress, August 28 - September 2, 2011, Milano, Italy, S. 9893-9898, 2011
[7] Obst, M.; Runde, St.; Wolf, G.; Urbas, L.: Integration requirements of package units - A description approach with FDI. In: Proceedings of the IEEE ETFA'13, doi:10.1109/ETFA.2013.6647974, 2013
[8] Obst, M.; Hahn, A.; Urbas, L.: Package Unit Integration for Process Industry - A New Description Approach. In: Proceedings of the IEEE ETFA'14, doi:10.1109/ETFA.2014.7005159, 2014
[9] John, D.; Danzer, B.; Riedl, M.; Zipper, H.: Selbsterklärende Geräte. Abbildung semantischer Information auf Parameter mit EDD. atp edition – Automatisierungstechnische Praxis 55(10), S. 56-61, 2013
[10] Diedrich, Ch.; Meyer, M.; Evertz, L.; Schäfer, W.: Dienste in der Automatisierungstechnik. Automatisierungsgeräte werden I40-Komponenten. atp edition – Automatisierungstechnische Praxis 56(12), S. 24-35, 2014
[11] Arifulina, S.; Walther, S.; Becker, M.; Platenius, M.C.: SeSAME: Modeling and Analyzing High-Quality Service Compositions. In: Proc. 29th ACM/IEEE Int. Conf. Automated Software Engineering, S. 839-842, doi: 10.1145/2642937.2648621, 2014
[12] Schüller, A.; Scholz, A.; Tauchnitz, Th.; Drath, R.; Scherwietes, Th.: Speed-Standardisierung am Beispiel der PLT-Stelle. Datenaustausch mit dem Namur-Datencontainer. atp edition – Automatisierungstechnische Praxis 57(1-2), S. 36-46, 2015
[13] OASIS: OASIS Reference Architecture Foundation for Service Oriented Architecture Version 1.0; http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/cs01/soa-ra-v1.0-cs01.pdf; zuletzt 20.01.2015
[14] PLI Development Guide, GE Fanuc International, 2003, http://support.ge-ip.com/support/resources/sites/GE_FANUC_SUPPORT/content/live/DOCUMENT/0/DO14/en_US/PLI%20Development%20Guide.pdf, zuletzt: 22.01.2015
[15] EClass: Classification and Product Description; http://www.eclass.de/; zuletzt: 20.01.2015
[16] QUDT: Quantities, Units, Dimensions and Data Types Ontologies; http://www.qudt.org/; zuletzt: 20.01.2015
[17] VDI3699 Blatt 2: Prozessführung mit Bildschirmen– Grundlagen. Beuth Verlag, 2005
[18] Hoernicke, M.; Christiansen, L.; Fay, A.: Anlagentopologien automatisch erstellen. Modelle aus der Mensch-Maschine-Schnittstelle erzeugen. atp edition - Automatisierungstechnische Praxis 56(04), S. 28–40, 2014
[19] GraphML Team: The GraphML File Format. http://graphml.graphdrawing.org/, zuletzt: 20.01.2015
[20] Schmitz, St.: Grafik- und Interaktionsmodell für die Vereinheitlichung grafischer Benutzungsschnittstellen der Prozessleittechnik. Dissertation RWTH Aachen, VDI-Verlag 2010
[21] Urbas, L.; Hennig, St.; Hager, H.; Doherr, F..; Braune, A.: Towards context adaptive HMIs in process industries. In: Proc. 9th IEEE International Conference on Industrial Informatics, S. 244–249. IEEE 2011
[22] Reuther, A: UseML. Systematische Entwicklung von Maschinenbediensystemen mit XML. Dissertation Universität Kaiserslautern. PAK 2003
[23] Calvary, G.; Coutaz, J.; Thevenin, D.; Limbourg, Q.; Bouillon, L.; Vanderdonck, J.: A unifying reference framework for multi-target user interfaces. Interacting with Computers, 15(3), S. 289–308, 2003
[24] Hickson, I.; Berjon, R.; Faulkner, S.; Leithead, T.; Navara, E. D.; O'Connor, E.; Pfeiffer, S.: HTML5 - A vocabulary and associated APIs for HTML and XHTML, W3C Recommendation 28 October 2014.

AUTOREN

Dipl. Ing. **MICHAEL OBST** (geb. 1985) ist wissenschaftlicher Mitarbeiter der Professur für Prozessleittechnik an der Technischen Universität Dresden mit den Schwerpunkten: Informationsmodellierung, Unterstützungssysteme für modulares Engineering.

Institut für Automatisierungstechnik,
Technische Universität Dresden, D-01062 Dresden,
Tel. +49 (0) 351 46 33 21 62,
E-Mail: michael.obst@tu-dresden.de

Dipl.-Ing. **THOMAS HOLM** (geb. 1979) ist wissenschaftlicher Mitarbeiter an der Professur für Automatisierungstechnik der Helmut-Schmidt-Universität/Universität der Bundeswehr, Hamburg. Sein Forschungsschwerpunkt liegt im effizienten Engineering von Automatisierungssystemen flexibler Produktionsanlagen.

Institut für Automatisierungstechnik,
Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg,
Holstenhofweg 85, D-22043 Hamburg,
Tel. +49 (0) 40 65 41 33 27, E-Mail: thomas.holm@hsu-hh.de

Prof. Dr.-Ing. **ALEXANDER FAY** (geb. 1970) ist Professor für Automatisierungstechnik an der Fakultät für Maschinenbau der Helmut-Schmidt-Universität/Universität der Bundeswehr, Hamburg. Sein Forschungsschwerpunkt sind Beschreibungsmittel, Methoden und Werkzeuge für einen effizienten Entwurf von Automatisierungssystemen.

Institut für Automatisierungstechnik,
Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg,
Holstenhofweg 85, D-22043 Hamburg,
Tel. +49 (0) 40 65 41 27 19, E-Mail: alexander.fay@hsu-hh.de

Prof. Dr.-Ing. **LEON URBAS** (geb. 1965) ist Inhaber der Professur für Prozessleittechnik an der Technischen Universität Dresden. Seine Hauptarbeitsgebiete beim Engineering verteilter sicherheitskritischer Systeme sind Funktionsintegration, modellgetriebenes Engineering, Modularisierung, Informationsmodelle der Prozessindustrie und Middleware in der Automatisierungstechnik. Einen weiteren Schwerpunkt bildet die Gebrauchstauglichkeit von mobilen Informationssystemen für die Prozessindustrie, Analyse, Gestaltung und Bewertung von Alarmierungs- und Unterstützungssysteme sowie Methoden der Benutzermodellierung zur prospektiven Gestaltung von Mensch-Technik-Interaktion.

Institut für Automatisierungstechnik,
Technische Universität Dresden, D-01062 Dresden,
Tel. +49 (0) 351 46 33 96 14,
E-Mail: leon.urbas@tu-dresden.de

Dr.-Ing. **SVEN KREFT** (geb. 1982) studierte Naturwissenschaftliche Informatik mit der Fachrichtung Robotik an der Universität Bielefeld. Anschließend war er als wissenschaftlicher Mitarbeiter in der Fachgruppe Produktentstehung am Heinz Nixdorf Institut der Universität Paderborn beschäftigt. Er promovierte 2012 mit einem Beitrag zur effizienten Bildung geospezifischer Umgebungsmodelle für interaktive Fahrsimulationen. Heute ist er Software-Produktmanager bei WAGO Kontakttechnik.

WAGO Kontakttechnik GmbH & Co. KG,
Hansastraße 27, D-32423 Minden,
Tel. +49 (0) 571 88 77 76 86,
E-Mail: sven.kreft@wago.com

Dipl.-Ing. **ULRICH HEMPEN** (geb. 1964) ist Leiter des internationalen Key Account und Branchenmanagements für die Prozess-, Fertigungs-, Transport- und Schiffsindustrie der WAGO Kontakttechnik GmbH & Co. KG. Nach seinem Studium der Elektrotechnik war er zunächst verantwortlicher Produktmanager für intelligente Feldgeräte bei der Hartmann & Braun AG, Bereichsleiter für Systemtechnik bei der Endress + Hauser AG und geschäftsführender Gesellschafter der Endler & Kumpf GmbH (Distributor und Systemintegrator der Fertigungs- und Prozessautomation).

WAGO Kontakttechnik GmbH & Co.KG,
Hansastraße 27, D-32423 Minden,
Tel. +49 (0) 571 88 73 80,
E-Mail: ulrich.hempen@wago.com

Dr. rer. nat. **THOMAS ALBERS** (geb. 1963) ist Geschäftsleiter Automation der WAGO Kontakttechnik GmbH & Co. KG. Nach seinem Studium der Diplomphysik mit anschließender Promotion war er Entwicklungsleiter bei Specs in Berlin, anschließend Entwicklungsleiter bei Baumüller Nürnberg Electronic und ist heute verantwortlich für Entwicklung, Produktmanagement und Marketing der Automatisierungsprodukte bei WAGO.

WAGO Kontakttechnik GmbH & Co.KG,
Hansastraße 27, D-32423 Minden,
Tel. +49 (0) 571 88 71 32,
E-Mail: thomas.albers@wago.com